VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Analýza dat v rozsáhlých databází energometrických měření

Data Analysis in Extensive Databases of Energometry Measurement

2012 Tomáš Žydek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Tomáš Žydek**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Analýza dat v rozsáhlých databází energometrických měření
Data Analysis in Extensive Databases of Energometry Measurement

Zásady pro vypracování:

1. Energometrická měření.
2. Statistické metody analýzy dat.
3. Databázové nástroje.
4. Návrh softwaru pro analýzu dat.
5. Realizace softwaru.
6. Testování softwaru a výsledků analýzy.
7. Zhodnocení výsledků práce.

Seznam doporučené odborné literatury:

[1] MOHYLOVÁ, Jitka a Vladimír KRAJČA. *Zpracování signálů v lékařství.* [CD/ROM] Žilinská universita 2005. ISBN 80-8070-341-8.
[2] BRONZINO,Joseph D. et al. *The Biomedical Engineering Handbook.* 1st Edition. Boca Raton(USA):CRC Press, 1995. 2896 s. ISBN 0849383463.
[3] MARTINÍK, Karel. *Obezita, nadváha: od teorie k praxi.* 3. vyd. Hradec Králové: Garamon s.r.o., 2008. 151 s. ISBN 978-80-86472-37-9.
[4] PENHAKER, Marek, Martin IMRAMOVSKÝ a Petr TIEFENBACH. *Lékařské diagnostické přístroje-učební texty.* Vyd. 1. Ostrava:VŠB-Technická univerzita, 2004. ISBN 80-248-0751-3.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Prauzek, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně, uvedl jsem všechny literární prameny a publikace ze kterých jsem čerpal.

V Ostravě dne 22.1.2013

Podpis Tomáš Zýdek

Poděkování

Za vedení bakalářské práce a odbornou konzultaci děkuji svému vedoucímu práce panu Ing. Michalu Prauzkovi, Ph.D. Dále bych chtěl poděkovat panu Ing. Karlu Vlachovi za připomínky a návrhy při vyvíjení aplikace v programovacím jazyku C#.

Abstrakt

Cílem bakalářské práce je vytvořit statistický software pro nemocnici Hradec Králové. Tento software bude sloužit lékařům k posouzení medicínských dat pomocí statistických analýz.

Klíčová slova

C#, analýza rozptylu, analýza hlavních komponent, bazální metabolismus, respirační kvocient, přímá kalorimetrie, nepřímá kalorimetrie, databázový jazyk SQL

Abstract

The goal of this bachellor's thesis is to develop statistical software for hospital Hradec Kralove. This software will serve physicians to consider medical records by statistical analysis.

Key words

C#, Analysis of Variance, Principal Component Analysis, Basal Metabolism, Respiratio Quotient, Direct Calorimetry, Indirect Calorimetry, SQL Database Language

Seznam použitých symbolů a zkratek

C#	C(sharp), programovací jazyk
RQ	respirační kvocient
CO_2	oxid uhličitý
O_2	kyslík
ATP	adenotrifosfát
BMR	z angl. Basal Metabolic Rate – bazální metabolismus
BMR(ž)	bazální metabolismus u ženy
BMR(m)	bazální metabolismus u muže
ANOVA	z angl. Analysis of Variance – analýza rozptylu
PCA	z angl. Principal Component Analysis – analýza hlavních komponent
Zedgraph	knihovna pro vykreslování grafů v programovacím jazyku C#
Dynamic Data Display	knihovna pro vykreslování grafů v programovacím jazyku C#
MathNet	knihovna pro statistické výpočty v programovacím jazyku C#
MySQL	typ databáze od firmy Oracle
NASA	národní úřad pro letectví a kosmonautiku

Obsah

Úvod

Tato práce se zabývá analýzou dat v rozsáhlých databází energometrických měření. Cílem této bakalářské práce je ralizace programu (softwaru) pro nemocnici Hradec Králové. Jedná se o program, který je napsán ve vývojovém prostředí Microsoft Visual Studio 2010 za použití programovacího jazyka C# (c-sharp). Tento statistický program, má za úkol umět pracovat se zdravotnickou databází, z níž má číst zdravotnická data, která bude program používat k vytvoření statistických grafů. Z těchto statistických grafů a výsledků sožitějších analýz může lékař posoudit, zda některý z pacientů se nepohybuje v mezích abnormálních hodnot a pokud ano, může přijít na to, proč tomu tak je a navrhnout řešení.

Kapitola 1
Energometrická měření

Přeměna látek a energie (metabolismus) patří k základním charakteristikám života. V organismu probíhají anabolické procesy spotřebovávající energii, které mají syntetickou povahu (růst, obnova tkání) a katabolické procesy, při nichž štěpením vznikají produkty za současného uvolňování energie. Organismus získává energii z potravy oxidací živin, přičemž vodík a uhlík se váže s kyslíkem na vodu a oxid uhličitý. Získanou energii může převádět na jiné formy energie, avšak vždy s určitou ztrátou. Část energie je v těle zachycována ve formě makroergních fosfátových vazeb a část energie se mění v teplo. Látková výměna, která pokrývá základní vitální funkce organismu, se nazývá bazální metabolismus. Organismus také využívá energii pro primární transport iontů a organických látek přes buněčné membrány, proteosyntézu (tvorba bílkovin za předpokladu dostatečného přísunu aminokyselin a energie), tvorbu tepla a svalovou kontrakci. Jako zdroje energie využívá cukrů, tuků a bílkovin [1].

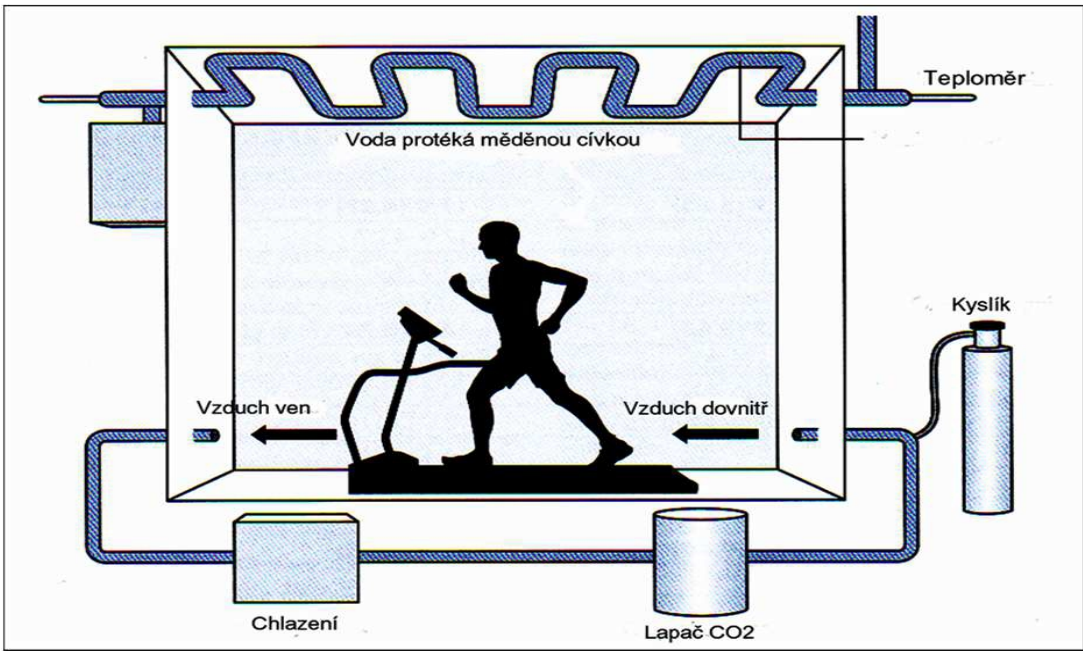
Měření energetického výdeje se provádí například pomocí následujících metod:

- Přímá kalorimetrie
- Nepřímá kalorimetrie
- Dvojitě značená voda
- Monitoring srdeční frekvence
- Sebepozorování – sledování fyzické aktivity a příjmu jídla

[2]

1.1 Přímá kalorimetrie

Metoda přímé kalorimetrie je založena na principu měření vyprodukovaného tepla. Pacient je ve speciální uzavřené místnosti, kde množství vyprodukovaného tepla pacientem je převedeno na množství spálených kalorií. Tato metoda je drahá, protože k její realizaci je zapotřebí mít speciálně vybavenou místnost pro tento druh měření. Proto se tato metoda v běžné technické praxi nevyužívá. Lze se s ní setkat pouze na některých specializovaných pracovištích ve vědecké praxi.



Obrázek 1.1: Místnost s vybavením pro realizaci metody přímé kalorimetrie [2]

1.2 Nepřímá kalorimetrie

V klinické a půmyslové praxi používáme pro stanovení vellikosti energetického výdeje metodu nepřímé kalorimetrie. Tato metoda je založena na poznání, že celkové množství energie, uvolněné na vykonávanou práci z makroergních fosfátových vazeb, musí být v konečné fázi uhrazeno oxidativním štěpením živin. Výpočet se provádí z množství spotřebovaného kyslíku na pracovní operaci, který násobíme energetickým ekvivalentem. Tento energetický ekvivalent stanovíme výpočtem, nebo z tabulek pro zjištěnou hodnotu respiračního kvocientu. Spotřeba kyslíku zjištěná během rovnovážného stavu nese informaci o velikosti energetického výdeje při lehkém a středním pracovním zatížení. Informaci o velikosti energetického výdeje při práci těžké pak nese součet spotřeby kyslíku měřené jak v pracovní, tak během celé zotavné fáze [1].

Energetický ekvivalent jednoho litru kyslíku je 20,2KJ (4,82Kcal). Tento energetický ekvivalent je stabilní za předpokladu smíšené stravy obsahující směs sacharidů, tuků a proteinů [1].

1.2.1 Respirační kvocient (RQ)

Při štěpení cukrů se uvolní právě tolik molekul CO_2, kolik se spotřebuje molekul kyslíku. Při spalování tuků vstupuje do reakcevíce molekul kyslíku než je počet vzniklých molekul CO_2. Poměr mezi CO_2 a O_2 ve vydechovaném vzduchu za jednotku času při ustáleném stavu se nazývá respirační kvocient. Následující tabulka ukazuje, jak se respirační kvocient mění v závislosti na spalované složce potravy [1].

Složka potravy	**Respirační kvocient**
Tuky	0,7
Bílkoviny	0,82
Sacharidy	1

Tabulka 1.1: Hodnoty respiračního kvocientu

Bílkoviny slouží jako zdroj energie jen v omezeném rozsahu. Jejich hlavní význam je v tom, že poskytují materiál pro růst a regeneraci organismu. Bezdusíkaté zbytky aminokyselin se metabolizují stejnou cestou, jako uhlohydráty. Jejich respirační kvocient leží zhruba uprostřed mezi sacharidy a tuky. Při průmyslových měřeních se vliv bílkovin na výsledném resporačním kvocientu a energetický ekvivalent zanedbává. To ovšem neplatí pro některé jiné látky, které mohou být metabolizovány jako např. alkohol, jehož spalné teplo leží mezi sacharidy a tuky, jeho respirační kvocient je 0,67. Chemická rovnice 1.1 popisuje spalování alkoholu v těle, rovnice 1.2 pak ukazuje respirační kvocient při spalování alkoholu organismem [1].

$$2\,C_2H_5OH + 6\,O_2 \leftrightarrow 4\,CO_2 + 6\,H_2O \qquad (1.1)$$

$$RQ \leftrightarrow \frac{4}{6} \leftrightarrow 0{,}67 \qquad (1.2)$$

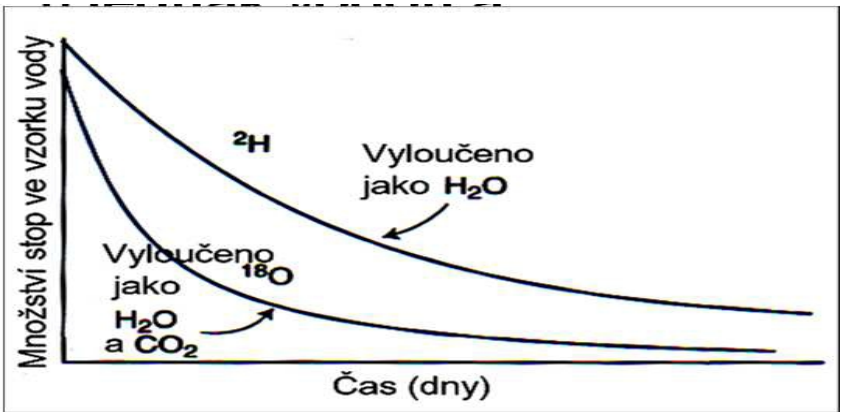
Výsledný respirační kvocient vyjadřuje poměr v jakém jsou spalovány především sacharidy a tuky. Je-li známo množství energie, která se uvolní při spotřebě jednoho litru kyslíku při spalování čistě sacharidů, nebo čistě tuků, můžeme z výsledného respiračního kvocientu vypočítat množství energie, která se uvolnila při konkrétní hodnotě respiračního kvocientu. Množství energie, které se uvolní při spotřebě jednoho litru kyslíku, se označuje jako energetický ekvivalent (E) [1].

Obrázek 1.2: Měření metodou nepřímé kalorimetrie v praxi

1.3 Dvojitě značená voda

Tato metoda využívá k určení energetického výdeje rozdílu mezi přijatým a vyloučeným množstvím izotopů vodíku, deuteria nebo $^{2}H_2$ a kyslíku ^{18}O za jednotku času. Pacient vypije dané množství vody s přesným obsahem izotopů, které se po několika hodinách rovnoměrně distribuují v tělesných tekutinách. Značkovaný $^{2}H_2$ vodík postupně opouští organismus především močí a potem. ^{18}O je vylučován jako součást vody a také jako zplodina metabolismu CO_2. Z rozdílu rozsahu eliminace těchto izotopů v daném čase lze vypočíst produkci množství CO_2. Poté ze známého nebo odhadovaného respiračního kvocientu (poměr mezi vyprodukovaným CO_2 a spotřebovaným O_2 za jednotku času) se přibližně vypočte spotřeba O_2 a z ní se pak stanoví hodnota energetického výdeje pacienta [3].

Metoda dvojitě značené vody se hojně nevyužívá, neboť k vyhodnocení rozdílů izotopů je zapotřebí použít hmotnostní spektrometr. Nutnost použít hmotnostní spektrometr dělá tuto metodu nákladnou, né každé pracoviště je jím vybaveno. Tato metoda je vhodná pro dlouhodobé sledování.



Obrázek 1.3: Graf postupného vylučování „značené“ vody z organismu [2]

1.4 Monitoring srdeční frekvence

Měřením energerického výdeje pacienta za pomocí monitoringu srdeční frekvence se diagnostik dopustí značné chyby a variability ve výsledcích měření. Srdeční frekvence je přímoúměrná spotřebě kyslíku pouze při submaximálním zatížení. Z tohoto poznatku vyplývá, že měření energetického výdeje u tepových frekvencí velmi nízkých nebo velmi vysokých dojde ke zkreslení výsledků měření. Spotřeba kyslíku organismem je dále ovlivněna teplotou, vlhkostí prostředí, nadmořskou výškou, současném zatížení organismu a také na jeho psychickém stavu [2].

Při srovnávání 24-hodinového energetického výdeje u 22 pacientů metodou kalorimetrie a metodou srdeční frekvence se zjistilo, že odchylky energetického výdeje mezi oběma metodami se pohybují od +20% do -15% [2].

Obrázek 1.4: Beurer PM58 – Hodinky s kardiomonitorem

1.5 Sebepozorování

Technika sebepozorování spočívá v tom, že si pacient sám hlídá svoji fyzickou aktivitu a příjem potravy. Všechnu tuto aktivitu si pacient zapíše do speciálního protokolu ze kterého se vždy na konci dne vypočítá přibližná energetická hodnota kterou pacient získal potravou a kterou vydal při různých denních aktivitách. Takto získaná hodnota denního energetického výdeje je pouze hodnotou orientační a od skutečných dohnot energetického výdeje se může značně lišit.

1.6 Základní pojmy k výpočtu energetického výdeje

Energie v potravinách je uložena v chemických vazbách mezi různými molekulami. Štěpení těchto vazeb uvolňuje energii. Například glukóza je rozložena v procesu glykolýzy na ATP, která je dále transformována na energii mechanickou – pohyb [1].

1.6.1 Bazální metabolismus (BMR)

Látková výměna, která pokrývá základní vitální funkce organismu se nazývá bazální metabolismus (BMR – z angl. Basal Metabolic Rate). Měří se v leže ráno nalačno nebo se odečítá z tabulek pro příslušné pohlaví, věk a velikost plochy těla. Bazální metabolismus je závislý na pohlaví (muži mají BMR o cca 10% vyšší), věku (s věkem klesá) a velikosti organismu. Bazální metabolismus vztažen na jeden kilogram hmotnosti těla je u různých druhů savců různý, v přepočtu na m^2 se však liší jen nepatrně. Tělesná teplota zvyšuje BMR o 14% na 1°C. Trávení zvyšuje BMR u cukrů a tuků o 4-6%, u bílkovin až o 30%. Průměrná hodnota BMR u dospělého člověka činí 105kJ na kilogram a den. Jakákoliv fyzická aktivita pak zvyšuje energetický výdej až na několikanásobky bazálního metabolismu [1].
Bazální metabolismus vyjadřuje množství energie, které organismus spotřebuje pouze svými vnitřními metabolickými procesy. Patří zde i energie, kterou spotřebují životně důležité orgány jako srdce, plíce a mozek pro svoji činnost. Každá další vykonávaná činnost organismem (chůze, běh, práce na počítači) spaluje energii navíc. Tabulka 1.1 ukazuje kolik energie organismus spálí za hodinu vykonánavé činnosti při různém fyzickém zatížení [1].

Práce	**kJ/h**
Velmi lehká	628
Lehká	628 - 1255
Středně těžká	1255 – 1883
Těžká	1883 - 2510

Tabulka 1.2: Průměrný výdej energie za jednu hodinu svalové činnosti [1]

1.6.2 Výpočet bazálního metabolismu

Vzorec pro výpočet bazálního metabolismu byl vynalezen z důvodu jeho náročného měření, zejména z důvodu dodržení přísných podmínek pro správnost měření. Bazální metabolismus se měří brzy ráno po probuzení, nejméně osmi hodinách revitalizace (spánku) a dvanácti hodinách po jídle. Je také nutno zajistit, aby nervový systém pacienta nebyl po dobu měření stimulován a teplota prostření aby odpovídala fyziologickým hodnotám. Tyto podmínky jsou zpravidla velkou překážkou pro provedení měření. Málokterý pacient bude souhlasit setrvat v nemocničním zařízení přes noc. První významná rovnice pro výpočet bazálního metabolismu vytvořili již v roce 1918 James Harris a Francis Benedict. Před vytvořením rovnice následovalo měření 136 mužů, 103 žen a 94 novorozených kojenců. Osoby byly vybrány tak, aby každá z nich odpovídala reprezentativním hodnotám populace co se týče věku, výšky a váhy. Poté na základě naměřených výsledků a za použití statistiky byly vytvořeny rovnice pro výpočet bazálního metabolismu u žen a mužů jak udávají vzorce 1.3 a 1.4 [4], [5].

$$BMR(\check{z}) = 655{,}0955 + (9{,}5634 \times váha\ v\ kg) + (1{,}8496 \times výška\ v\ cm) - (4{,}6756 \times věk) \quad (1.3)$$

$$BMR(m) = 66{,}473 + (13{,}7516 \times váha\ v\ kg) + (5{,}0033 \times výška\ v\ cm) - (6{,}755 \times věk) \quad (1.4)$$

Například muž, 22 let, 75 kg, 195 cm vysoký má bazální metabolismus pomocí Harris-Benedictovy rovnice 1925 kcal. Tuto energii jeho tělo využije pro zachování všech vitálních funkcí organismu za den, avšak za předpokladu, že jeho tělo nebude vykonávat žádnou práci, ani pohyb. Pro zjištění energie, kterou bude organismus potřebovat při různém pracovním zatížení za den lze využít Harris-Benedictovy tabulky, viz. Tabulka 1.3.

Práce (četnost za týden)	**Potřebné kalorie za den**
Žádná až lehká	BMR × 1,2
Lehká (1-3 dny)	BMR × 1,375
Střední (3-5 dnů)	BMR × 1,55
Těžká (6-7 dnů)	BMR × 1,725
Velmi těžká (2x za den, 7 dnů v týdnu)	BMR × 1,9

Tabulka 1.3: Harris-Benedictův prinicp spotřeby energie za den [4], [5]

Dle výše uvedené tabulky 1.3 lze vypočítat, že jedinec s bazálním metabolismem 1925 kcal a středním pracovním zatížením bude potřebovat denně příjmout 2984 kcal aby si zachoval svou váhu 75 kg. Může se zdát, že je to hodně kalorií, ale jeho tělo vzhledem k pracovnímu vytížení to vyžaduje.

1.6.3 Pracovní metabolismus

Jedná se o klidový metabolismus navýšený o energii využitou během pracovních aktivit. Jakákoliv zyzická aktivita zvyšuje energetický výdej až na několikanásobky bazálního metabolismu. Lehká svalová činnost (např. sezení, stání, mírná chůze) zvyšuje bazální metabolismus o 25-60 %, středně těžká svalová práce (např. rychlá chůze, plavání, práce v domácnosti) o 100 % a více, těžká svalová práce (horníci, vrcholový sport) o 600-1500 % [1].

1.7 Vyhodnocení energetického příjmu

V praxi pro vyhodnocení energerického přijmu je možné počítat s převodním vztahem, že jedna kilokalorie je úměrná 4,2 kJ. Dále pak lze počítat s gramy dané živiny, kde:

- 1 g sacharigů = 17 kJ = 4 kcal
- 1 g proteinů = 17 kJ = 4 kcal
- 1 g bílkovin = 38 kJ = 9 kcal

Jelikož různé energetické tabulky uvádějí různé energetické hodnoty jednotlivých potravin, může být toto počítání kalorií velmi nepřesné. Energetický obsah v potravině může být různý v závislosti na přípravě potraviny. Je nutno také brát v úvahu, že čím více vlákniny má organismus k dispozici, tím méně kalorií tělo vstřebá [2].

Kapitola 2
Statistické metody analýzy dat

Jedním ze základních předpokladů použití matematické statistiky pro eliminaci chyb měření, vzniklých působením náhodných vlivů, je možnost pořízení souboru dostatečného počtu pozorovaných (měřených) hodnot sledované veličiny za reprodukovatelných podmínek. Tento požadavek může být dominantním problémem při řešení konkrétních provozních měření. Nesplnitelnost předpokladů vede v praktických případech k pořízení datových souborů, jejichž vlastnosti nezaručují korektnost použití metod statistické analýzy a vedou k získání výsledků, které jsou v rozporu se skutečností [6].

2.1 Analýza jednorozměrných dat

Při instrumentálních měřeních získáváme náhodný výběr dat, jehož prvky (jednotlivá měření, pozorování) jsou uvažovány jako realizace určité náhodné veličiny. Podstatné je získat tzv. reprezentativní náhodný výběr, který je základním předpokladem korektnosti při použití statistických metod pro vyhodnocení výsledků měření [6].
Reprezentativnost analyzovaných dat je zajištěna způsobem, jakým byly data naměřeny. Každý pacient, jehož data jsou k dispozici ve zdravotnické databázi prošel naprosto stejným průbehem měření, který je shodný se zdravotnickými standardy prováděných úkonů. Analýzou jednorozměrných dat zdravotnické databáze se rozumí použití statistických úkonů jako zjištění střední hodnoty, modusu, mediánu, minima a maxima z výběrového souboru dat. Tato statistická analýza dat významně přispěje k hodnocení medicínských závěrů pro vybranou skupinu pacientů.

2.1.1 Modus

Nejčastěji se vyskytující hodnota v datovém souboru se nazývá modus. Tato metoda patří mezi odhady parametrů charakteristik polohy. Výhodou je, že jí lze použít i na datové soubory nečíselných typů, tj. slovní datové soubory. Nechť datový soubor „X“ obsahuje hodnoty {1, 2, 3, 2, 2, 6}, poté modus daného datového souboru bude číslo 2. Modus odpovídá maximální hodnotě na křivce rozložení hustoty pravděpodobnosti. Křivku rozložení hustoty pravděpodobnosti lze nahradit histogramem.

2.1.2 Medián

Tato metoda také patří mezi odhady parametů charakteristik polohy. Medián, je taková hodnota z datového souboru, která má nad sebou a pod sebou 50% hodnot z datového souboru. Jedná se tedy o padesátiprocentní kvantil z datového souboru. Nechť datový soubor „Y“ obsahuje hodnoty {1, 2, 3, 4, 5}, poté median daného datového souboru bude číslo 3, které odpovídá podmínce, že existuje 50% hodnot nad a 50% hodnot pod mediánem. Obsahuje-li datový soubor sudý počet prvků, pak medán bude aritmetický průměr z dvou prostředních prvků datového souboru. Metodu medián lze použít i pro slovní datové soubory, avšak pouze pro takové datové soubory, které obsahují lichý počet prvků. Pro jednoduché určení mediánu je výhodné datový soubor seřadit. Medián se poté nachází uprostřed datového souboru.

2.1.3 Střední hodnota (aritmetický průměr)

Také se jedná o odhady parametrů charakteristik polohy. Střední hodnota (aritmetický průměr) je součet všech hodnot prvků datového souboru podělený počtem prvků datového souboru. Jedná se o nejčastěji používanou charakteristiku polohy. Střední hodnota vypovídá o tom, jaká hodnota připadá na jednu naměřenou hodnotu z datového souboru. Střední hodnota je vysoce ovlivněna extrémně vybočujícími hodnotami v datovém souboru. Proto pro její přesnou interpretaci je vhodné odstranit z datového souboru extrémně vybočující hodnoty. Je třeba tedy dosáhnout stejnorodého datového souboru.

2.1.4 Minumum a maximum

Minimum reprezentuje vždy nejnižší hodnotu nacházející se v datovém souboru. Naopak maximum představuje vždy největší hodnotu z datového souboru.

2.1.5 Kvantily

Jedná se o zváštní druhy číselných charakteristik polohy. Jsou-li prvky datového souboru seřazeny sestupně, pak lze říci, že dvacetiprocentní kvantil daného datového souboru je taková hodnota, která má pod sebou dvacet procent zbývajících hodnot datového souboru. Stoprocentní kvantil, tedy představuje nejvyšší hodnotu ze souboru dat.

2.2 Analýza dvourozměrných dat

Kromě testů analýzy jednorozměrných dat je velmi přínosem zjistit, jak se pohybují ostatní naměřené hodnoty a zda mají nějakou spojitost vzhledem k výsledkům základní analýzy jednorozměrných dat. Tyto spojitosti bude mít za úkol vypočítat software, který je výsledkem bakalářské práce. Na lékaři poté bude, aby na základě znalostí z medicíny a statistických poznatků zhodnotil, zda-li pak šetřená spojistost má nějaký význam, či nikoliv.
Například lékař vybere z databáze pacienty, kteří váží od 85 kg do 90 kg a bude chtít u této skupiny pacientů sledovat hladinu cukru v krvi a to hodinu po podání glukózy na lačno. Statistickými úkony může poté lékař zjistit abnormální chování organismu u zkoumané skupiny pacientů a dále může zkoumat, proč tato abnormalita nastala, popřípadě stanovit řešení.

2.3 Analýza rozptylu (ANOVA)

Anova je technika umožňující posouzení jednotlivých zdrojů variability v datech. U opakovaných měření existují vždy nějaké odchylky. Tyto náhodné odchylky mohou způsobit, že se obtížně zjišťuje významnost rozdílů mezi skupinami paralélních měření. Základní myšlenkou analýzy rozptylu je v tomto případě, zda a jak může být v sadě výsledků paralélních (opakovaných) měření statisticky rozpoznáno rozdělení do skupin (např. podle vyšetřujícího lékaře, popdle postupu). Celkový rozptyl celé sady dat je dán kombinací rozptylů mezi skupinami a uvnitř skupin. Anova umožňuje separovat jednotlivé zdroje rozptylu a dílčí rozptyly vzájemně pozorovat za účelem určení, zda jsou rozdíly mezi nimi statisticky významné. Anova nám umožňuje odpovědět na otázku, zda jednotlivé skupiny reprezentují výběry z jednoho základního souboru. Analýza rozptylu je užitečná při analýze dat získaných při plánovaných experimentech. Ve srovnání s t-testem pro nezávislé střední hodnoty si anova vystačí s menším množstvím výpočtů [7].

2.3.1 Jednofaktorová analýza rozptylu

Jednofaktorová analýza rozptylu (one-way anova), anova s jednoduchým tříděním. Používá se tehdy, přichází-li v úvahu pouze jeden faktor, který nabývá několika hodnot a pro každou hodnotu máme skupinu paralélních stanovení. Jde o to, zda rozdíly mezi měřeními jsou statisticky významné nebo zda jsou pouze důsledkem běžných náhodných odchylek. Celková variabillita v datech je dána kombinací rozptýlení výsledků jednoho každého měření a rozptýlení mezi středními hodnotami různých výsledků [7].

2.4 Analýza hlavních komponent

Hlavním úkolem při analýze rozsáhlých souborů dat a jejich klasifikaci je nalezení lepší reprezentace pomocí vhodné transformace. Analýza hlavních komponent (PCA – Principal Component Analysis) je klasická metoda analýzy a komprese dat. Cílem této analýzy je nalézt na základě rozsáhlého počtu proměnných menší množinu nových proměnných s menší redundancí, která by poskytovala nejlepší možnou reprezentaci dat. Přitom redundance je měřena pomocí korelace mezi proměnnými. Odstranění redundance dat je stěžejní právě pro klasifikaci, kde nadbytečnost datového souboru může způsobit nečitelnost, nebo nemožnost klasifikace. Hlavní komponenty jsou umělé veličiny vytvořené jako lineární kombinace vstupních parametrů. Tyto komponenty lze vypočítat pomocí transformační matice [8].

2.5 Faktorová analýza

Podobně jako metoda hlavních komponent patří také faktorová analýza mezi metody redukce počtu původních proměnných. Ve faktorové analýze předpokládáme, že každou proměnnou můžeme vyjádřit jako lineární kombinaci nevellkého počtu polečných skrytých faktorů a jediného chybového faktoru. Na rozdíl od komponentní analýzy se při faktorové analýze snažíme vysvětlit závislost proměnných. K nevýhodám metody patří zejména nutnost zadat počet společných faktorů ještě před prováděním vlastní analýzy [7].

2.6 Porovnání faktorové analýzy a analýzy hlavních komponent

Obě metody jsou nepoužitelné, když jsou původní data nekorelované. Faktorová analýza nemá co objasnit a analýza hlavních komponent povede k hlavním komponentám totožným s původními proměnnými. Faktorová analýza se pokouší objasnit kovariance a korelace původních proměnných pomocí několika společných faktorů. Analýza hlavních komponent objasňuje pouze rozptyl původních proměnných. Když do analýzy hlavních komponent přidáme další proměnnou, původní komponenty se nezmění, kdežto když přidáme další faktor do faktorové analýzy, ostatní faktory se změní [7].

2.7 Využití statistických metod analýzy dat

Pro konečný výsledek bakalářské práce, což je navržený software pro analýzu medicínských dat je primární cíl umět základní analýzu jednorozměrných dat. Pomocí jednoduchých statistických úkonů jako jsou modus, medián, střední hodnota, minimum a maximum může lékař na základě medicínských zkušeností snadno odhalit různé abnormality u pacientů a může navrhnout řešení jak jim předcházet. Některé ze složitějších metod analýzy dat přesahují rámec bakalářské práce, avšak některá z nich bude součástí jejího řešení.

Kapitola 3
Databázové nástroje

Lidé již od pradávna evidují a uchovávají data. V době, kdy neexistovaly počítače se data zaznamenávala na papír, ale již v té době měla data struktůru nějaké tabulky. K uchovávání dat se používaly papírové kartotéky a operace s nimi prováděl člověk. S příchodem počítačů se data zaznamenávala fyzicky na pevný disk, avšak záznamy tohoto typu brzy vzrostly do velkých rozměrů. Proto se brzy začaly vyvíjet nástroje, které v dnešním světě známe pod pojmem databáze. Databáze umožňuje logické uspořádání databázových záznamů (dat), větší ochranu před její ztrátou. Její logické uspořádání je důležité pro rychlý přístup k datovým záznamům, také pro snažší a rychlejší výpočet funkcí pro třídění dat.

3.1 Entita, atribut, vazba mezi entitami

Databáze slouží k uchovávání dat, která popisují věci z reálného světa. Například evidence pacientů v nemocnici, evidence o prodaných kusech zboží, atd. Entita je považována za prvek z reálného světa. Příklady entit jsou: zástěra, město, člověk. Entity jsou popsány svými vlastnostmi – atributy. Příklady entit k výše uvedeným atributům jsou: barva, velikost, věk. Vazbou mezi entitami se rozumí, že jednotlivé entity mají mezi sebou nějakou vazbu. Rozlišujeme několik základních druhů vazeb.

Vazba typu 1:1 se rozumí, že entity mezi sebou mají pouze jednu vazbu. Příklad takové vazby je, že jeden člověk vlastní právě jeden občanský průkaz s unikátním identifikačním číslem. Přiřazené identifikační číslo charakterizuje právě jednoho člověka a nemůže se stát, že by stejné identifikační číslo bylo přiřazeno někomu jinému.

Vazba typu 1:N se rozumí, že entita jednoho druhu může mezi sebou mít více vazeb jiného druhu, ale stejného typu. Příklad takové vazby je, že jeden člověk vlastní více kreditních karet od různých bankovních účtů, ale tyto kreditní karty je oprávněn používat pouze jeden člověk – majitel karty.

Vazba typu M:N se rozumí, že vazba mezi entitami není nijak omezena. Příkladem takové vazby je, že jeden člověk si může koupit více automobilů, ale jeden automobil může být prodán více lidem. Tato vazba musí být při návrhu databáze rozepsána na dva vztahy typu 1:N.

3.2 Relační databáze

Základy pro relační databáze položil v roce 1970 anglický IT specialista Edgar Frank Codd. Jeho navržený model se do dnešních dnů využívá pro návrh a tvorbu relačních databází.

Základním stavebním pojmem je relace, kterou si lze představit, jako klasickou tabulku, skládající se ze sloupců a řádků. Jedna tabulka (entita) popisuje řadu vlastností (atributů), které se evidují o dané entitě.
Nechť existuje tabulka KREDITNI_KARTY, která popisuje v bankovním systému seznam kreditních karet vydaných bankou. Sloupce v tabulce (atributy) pak budou: Č_KARTY, Č_SMLOUVY, MAJITEL_JM, MAJITEL_PŘ, PLATNOST_OD, PLATNOST_DO, TYP_KARTY.
Relace odpovídá dané tabulce a prvkům relace odpovídají řádky tabulky (databázové záznamy). Soubor tabulek pak tvoří celou databázi, kterou si lze prohlídnout na relačním schématu databáze.

3.3 Jazyk SQL

Jazyk SQL se skládá z několika částí. Jedná se o složky programovacího jazyka pro operace, které jsou potřebné pro operace s databázemi:

1. Definování dat
2. Manupulace s daty
3. Dotazování na data
4. Správa dat

Aby bylo možné s databází pracovat, musí se zadefinovat její struktůra – tabulky v databázi. Pojmem definování dat se rozumí příkazy pro vytváření tabulky v databázi. Nejpoužívanějším příkazem definování dat je příkaz vytvoř tabulku „CREATE TABLE“. Obsahuje-li databáze tabulky, pak je za potřebí tyto tabulky naplnit daty. Možnost naplnit tabulku jednotlivými záznamy umožňují příkazy pro manipulaci s daty. Příkladem příkazu pro manipulaci s daty mohou být příkazy „INSERT“ a „DELETE“. Existují-li v databázi tabulky naplněné jednotlivými záznamy, pak pomocí dotazovacího jazyka je možnost zobrazit takové záznamy z tabulky, které jsou specifické nějakým parametrem. Příkladem mohou být příkazy „SELECT“ a „BETWEEN“. Pomocí těchto příkazů je možnost zobrazit například všechny knížky autora, které napsal v posledních dvou letech. Typickým příkladem příkazu pro správu dat je příkaz „UPDATE“, pomocí kterého lze přepisovat hodnoty v tabulkách záznamů.

3.3.1 Příklad jazyka definování dat

CREATE TABLE Student (ID Integer NOT NULL, Jméno Varchar(15), Příjmení Varchar(20), PRIMARY KEY(ID))

Vytvoří tabulku „Student“ se sloupci „ID“, „Jméno“, „Příjmení“. Primární klíč je nastaven na sloupec ID, protože tento sloupec je pro tuto tabulku unikátním rozpoznávacím znakem. Platí zde, že jeden student má právě jeden rozpoznávací, unikátní ID znak. Do sloupce ID bude možno zapisovat pouze číselné hodnoty, do sloupce „Jméno“ bude možné zapsat pouze text o velikosti patnácti znaků a do sloupce „Příjmení“ bude možno zapsat text o velikosti dvaceti znaků.

3.3.2 Příklad jazyka pro manipulaci s daty

INSERT INTO Student (ID, Jméno, Příjmení) VALUES (1, "Natalie", "Portman")

Vloží do tabulky „Student“ hodnoty. Do sloupce „ID“ vloží číslo „1“, do sloupce „Jméno“ vloží jméno „Natalie“ a do sloupce „Příjmení“ vloží příjmení „Portman“. Při zapisování do výše vytvořené tabulky není třeba zadávat výčet prvků z tabulky. Obsahovala-li by tabulka „Student“ navíc atribut „ROD_Č“, pak je nutno vypsat včechny atributy do kterých se zapisuje. Musí se však dbát na to, aby nový záznam v tabulce obsahoval všechny povinné údaje. Povinným údajem v tabulce „Student“ je však pouze sloupec „ID“, takže použitým příkazem „INSERT“ se naplnily všechny potřebné údaje.

3.3.3 Příklad jazyka pro dotazování na data

SELECT * FROM Student WHERE ID =1

Příkaz „SELECT * “ vybere z tabulky „Student“ všechny sloupce které obsahuje, ale vypíše pouze záznam (řádek), který má hodnotu ID = 1. Záměnou znaku „* “ výčtem sloupců zobrazí pouze výčty daných sloupců, pro které platí zadaná podmínka.

SELECT * FROM Student WHERE ID BETWEEN >=1 AND <=5

Vybere z tabulky „Student“ všechny sloupce které jsou k dispozici, ale vypíše pouze řádky, které jsou větší, nebo rovno 1, ale menší, nebo rovno 5. Výsledek takového příkazu tedy bude zobrazení pouze takových řádků, které odpovídají zadané podmínce.

3.3.4 Příklad jazyka pro správu dat

UPDATE Student SET Příjmení=‘Croft‘ WHERE ID=1

Tento příkaz aktualizuje záznam v tabulce „Student“, sloupec „Příjmení“ kde ID studenta je rovno 1. Při nespecifikaci příkazu „WHERE“ bychom aktualizovali příjmení u všech záznamů v tabulce, což by bylo nežádoucí. Z tohoto důvodu je vždy nutné uvést, jaký řádek tabulky se přepisuje.

3.4 Systémy řízení báze dat

Systémy řízení báze dat se rozumí softwarové vybavení, které tvoří rozhraní mezi obsluhovým programem a uloženými daty v databázi.Systémy řízení báze dat spolu s daty tvoří databázový systém. Systémy řízení báze dat musí být schopny pracovat s velkými objemy dat, musí umět k nim efektivně přistupovat, vyhledávat v nich, modifikovat záznamy a také musí umět definovat strukturu těchto dat.

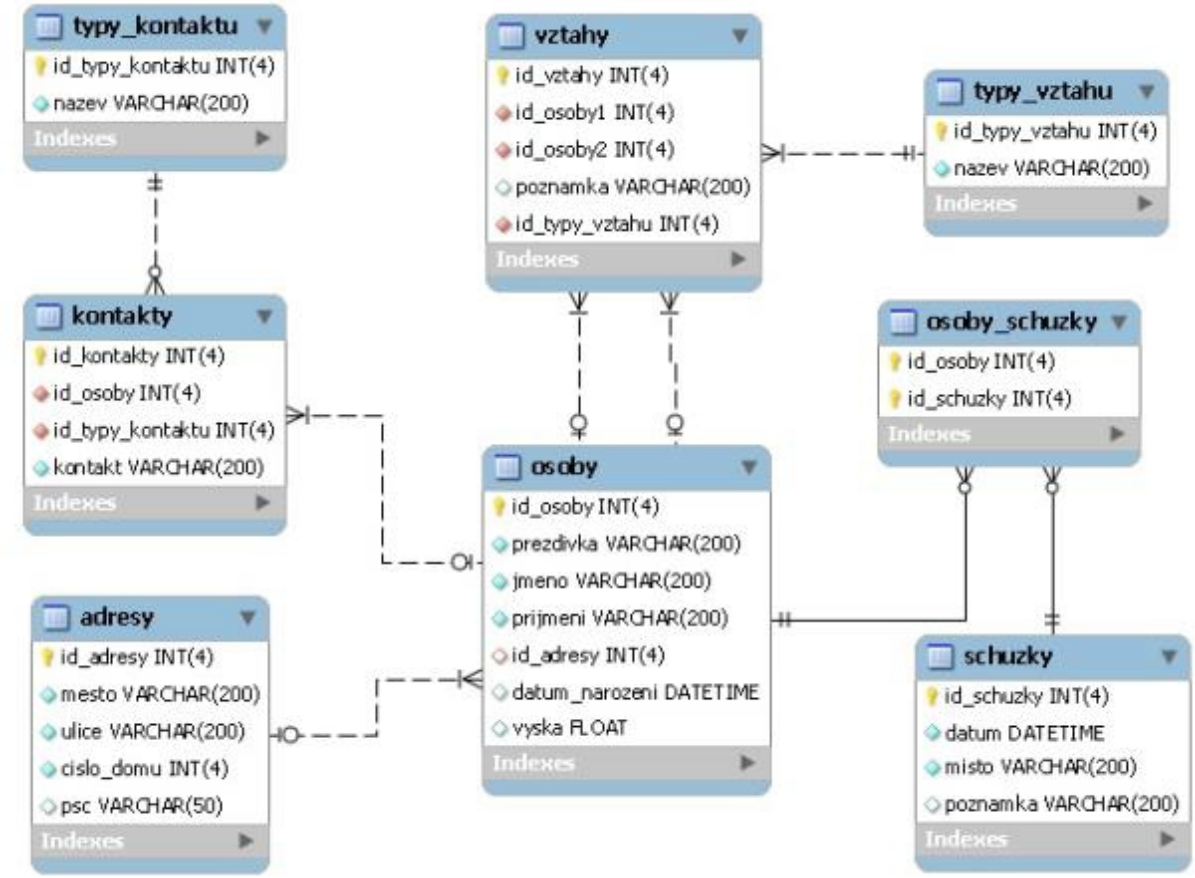
3.4.1 Nejznámější systémy řízení báze dat

Nejznámější systémy řízení báze dat jsou vypsány v následující tabulce.

Název	**Vývojový tým**
Oracle	Oracle Corporation
Firebird	Firebird Foundation
Microsoft SQL Server	Microsoft
MySQL	Oracle Corporation
PostgreSQL	PostgreSQL Global Development Group

Tabulka 3.1: Nejznámější systémy řízení báze dat

3.5 Příklad databázového schématu



Obrázek 3.1: Příklad databázového schématu [9]

Kapitola 4
Návrh softwaru

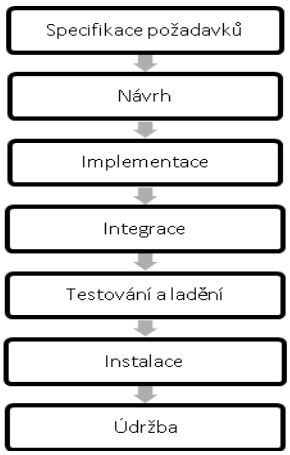
Při návrhu softwarového řešení, je velice důležité rozplánovat si jednotlivé fáze vývoje. Pro řádné rozplánování jednotlivých fází vývoje softwaru existuje celá řada modelů pro návrh a realizaci softwaru. O některých modelových typech návrhu softwaru bude tato kapitola pojednávat. Je také důležité si již v počátečním stádiu navrhování uvědomit, jaký bude funkční cíl softwaru, kdo jej bude obsluhovat a jakým způsobem budou zobrazovány výsledky, které program vyhodnotí. Důležitým poznatkem při vylepšování programu, je zpětná vazba od uživatele. Z tohoto důvodu je zřejmé, že navržený software je nutné neustále vylepšovat a že první verze softwaru nebude vždy to, co uživatel od navrhovaného softwaru čeká.

4.1 Modelové typy návrhu softwaru

Následující modelové situace poukazují na to, jak postupovat při návrhu softwaru a to od jeho počátku až po zavedení softwaru do ostrého provozu. Využití těchto modelových situací pomáhá vývojařům softwaru předejít chybám a odhalit je již v počáteční fázi vývoje. Chyby softwaru ve fázi jejího zavedení do provozu jsou nepřípustné a mohou mít katastrofální následky.

4.1.1 Vodopádový model

Tento model jako první popsal Winston W. Royce. Ačkoliv ve svém článku popsal tento model, jako model nefungující, stal se tento model jako výchozí pro vývojařské velmoci jako jsou firmy pracující pro Ministerstvo obrany Spojených států nebo NASA [10].



Obrázek 4.1: Vodopádový model [10]

Základní myšlenkou tohoto modelu je postupné přecházení z jedné fáze do druhe a to způsobem, kdy jedna fáze musí být dokončena a dokonale vypracována, než se přejde na fázi další. Odpůrci tohoto modelu však poukazují na fakt, že není možné jednu fázi vypracovat natolik dokonale, aby odpovídala následným požadavkům. Je zřejmé, že když se objeví problém, je nutné se k nějakému bodu návrhu vrátit. Uživatel by však měl uvést veškeré požadavky na software, aby nedošlo k nedorozumění mezi vývojaři a požadavky uživatele na software.

4.1.2 Spirálový model

Spirálový model je založený na čtyřech fázích vývoje programu, kdy se tyto fáze neustále opakují až do doby, kdy je software vyvinut a připraven k použití. Na počátku spirálového cyklu se provede analýza problému a zhodnotí se. Dále se aplikuje řešení problému a připravuje se na řešení problému dalšího. Na konci každého z těchto čtyř cyklů se provádí rekapitulace navrženého řešení.

Obrázek 4.2: Spirálový model [11]

4.2 Návrh softwaru pro analýzu dat

Při návrhu jakéhokoliv softwaru je důležité vědět od zadavatele práce, jaké požadavky na daný software jsou vyvíjeny, jaké funkční vlastnosti musí software umět zpracovat a vyhodnotit. Cílem tohoto softwaru pro analýzu dat je vyhodnocovat zdravotnická data a provádět z těchto dat jejich statistickou analýzu. Software musí umět pracovat se zdravotnickou databází. Tuto databázi musí umět prohlížet a číst z ní požadovaná data. Slovem požadovaná se myslí, že software musí umět dovolit uživateli výběr takových dat z databáze, která uživatel požaduje analyzovat. V praxi to znamená, že program musí mít naimplementovanou funkci pro třídění a výběr dat. Dále z těchto výběrů dat pro statistickou analýzu musí tuto analýzu provést v požadovaném rozsahu. Rozsahem analýzy se rozumí provedení výpočtů pro základní statistickou analýzu. Program dle požadavků zadavatele musí umět z vybraného souboru dat vypočítat minimum, maximum, střední hodnotu, medián a další.

Program musí také umět zobrazit některé ze základních statistických grafů aby měl uživatel k dispozici také vizuální interpretaci dat. Požadavky softwaru na zobrazování statistických grafů jsou předmětem konzultace se zadavatelem – vedoucím bakalářské práce.

Vyvíjený software bude plnit funkci dolování dat ze zdravotnické databáze. Data z ní bude vyhodnocovat pomocí předem navržených funkčních algoritmů. Uživatel – lékař poté z výsledků těchto analýz může provést závěr, který povede ke kategorizování jednotlivých skupin pacientů. Pacienti v těchto skupinách budou zpravidla pacienti s abnormálními nebo jinak vybočujícími hodnotami, což povede ke zjišťování příčin, z jakého důvodu mají tito pacienti odlišné hodnoty od ostatních.

Grafická úprava i funkční požadavky budou postupně doplňovány v závislosti na požadavcích, které budoou na software vyvíjeny ze strany uživatele. Touto cestou lze předejít vytvážení nových prvků, které nebudou využity a naopak soustředění se na takové funkce programu, které budou pro uživatele zajímavé.

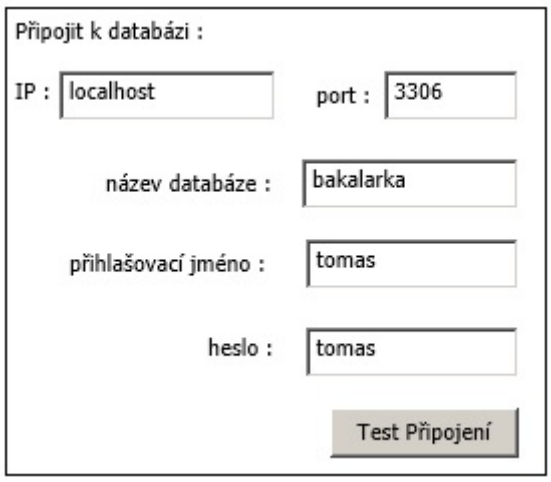
Kapitola 5
Realizace softwaru

Po návrhu softwaru a upřesnění veškerých parametrů, které musí program zvládat je vhodné přistoupit k jádru celé práce a to k realizaci softwaru. Původní myšlenka byla, že software bude napsán v prostředí „Matlab“, avšak z hlediska vize programu do budoucna je výhodnější napsat tento statistický software v některém z uznávaných programovacích jazyků. Z tohoto důvodu byl vybrán programovací jazyk C# a studentsky dostupný vývojový software „Microsoft Visual Studio“ od společnosti Microsoft. Programovací jazyk C# je lepší zejména z hlediska inovace programu do budoucna, která je plánována za předpokladu, že software bude mít v lékařském výzkumu využití.

5.1 Napojení na lékařskou databázi

Software využívá univerzálního přístupu napojení na jakoukoliv databázi, která je typu „MySQL“. Optimalizace softwaru na typ určité databáze bylo předmětem konzultace s vedoucím bakalářské práce a byla dohodnuta optimalizace na databázový typ „MySQL“, kterou vyvíjí specialisté z firmy Oracle.

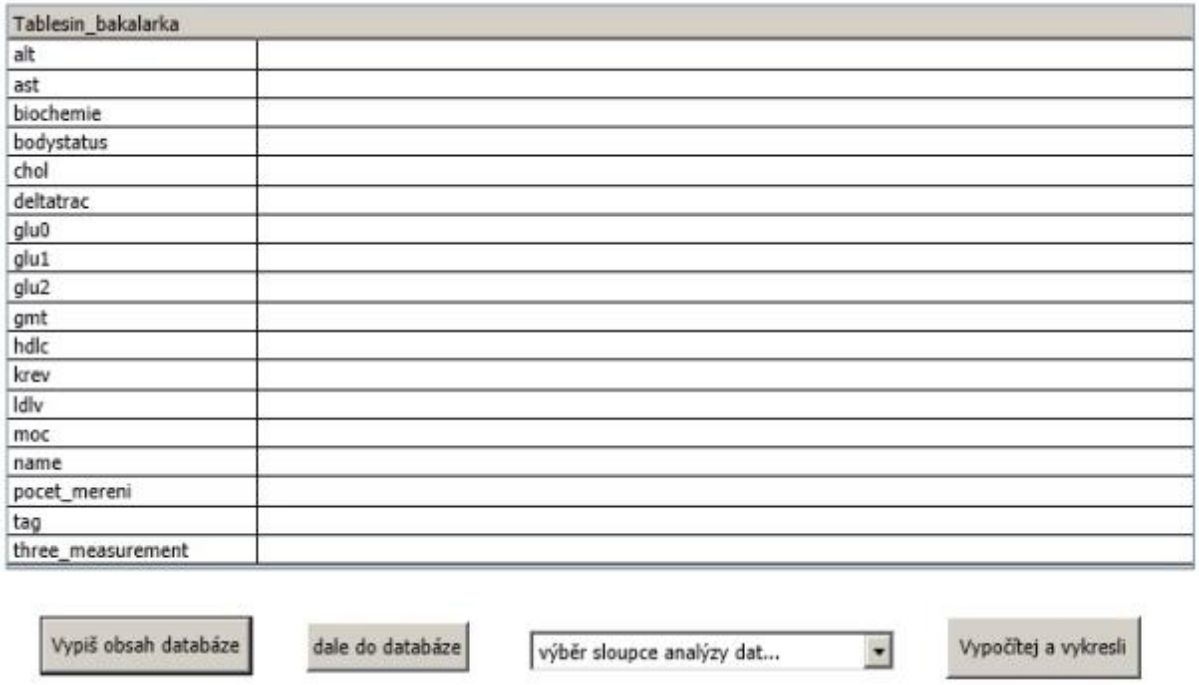
Databáze, na kterou se software napojuje může běžet přímo na počítači, z kterého je software spuštěn nebo se software může napojit i na vzdálenoý databázový server v internetu či intranetu. Důležité pro uživatele je vždy zadat správnou adresu IP databázového serveru a jeho port. Poté si uživatel musí být jist, že má příslušná práva pro vstup do databáze prostřednictvím svého uživatelského jména a hesla. Aby se software mohl bez problémů napojit, je nutné také znát jméno databáze, která běží na databázovém serveru (na kterou se chce uživatel napojit). Ovládací prvek pro připojení do databáze obsahuje také testovací tlačítko, které zjistí, zda je možno se na databázi napojit prostřednictvím zadaných údajů. Prvek lze vidět na následujícím obrázku.



Obrázek 5.1: Ovládací prvek pro připojení do databáze

5.2 Prohlížení databáze a ovládací prvky softwaru

Po zadání správných údajů pro přihlášení do databáze a stisknutí tlačítka „Vypiš obsah databáze“ se zobrazí právě obsah tabulek databáze. Obsah databáze se rozumí výčet včech tabulek, které databáze obsahuje. Pro vstup do těchto tabulek musí uživatel kliknou na příslušnou tabulku, do které chce vstoupit a následně stisknout ovládací tlačítko „dále do databáze“. Vstup do databázové tabulky uvádí do provozu roletkové menu „výběr sloupce analýzy dat“. Toto roletkové menu umožňuje uživateli vybrat data pro analýzu z takového sloupce, z jakého jí požaduje. Důležité je, aby uživatel v tabulce označil vždy takové řádky (pacienty), které chce analyzovat a následně vybral sloupec hodnot, které chce analyzovat. Je-li vše správně nastaveno, stiskem tlačítka „Vypočítej a vykresli“ se uživateli zobrazí veškeré výpočty a grafy, které software nabízí. Zobrazování a prohlížení tabulek databáze program zobrazuje v ovládacím prvku tabulkového zobrazení (datagridu). Vstoupí-li uživatel do nějaké tabulky v databázi, může se kdykoliv vrátit na obsah tabulek opětovným stisknutím tlačítka „Vypiš obsah databáze“. Následující obrázek ukazuje, jak vypadá výpis tabulek zdravotnické databáze, která je k dispozici pro testování softwaru.

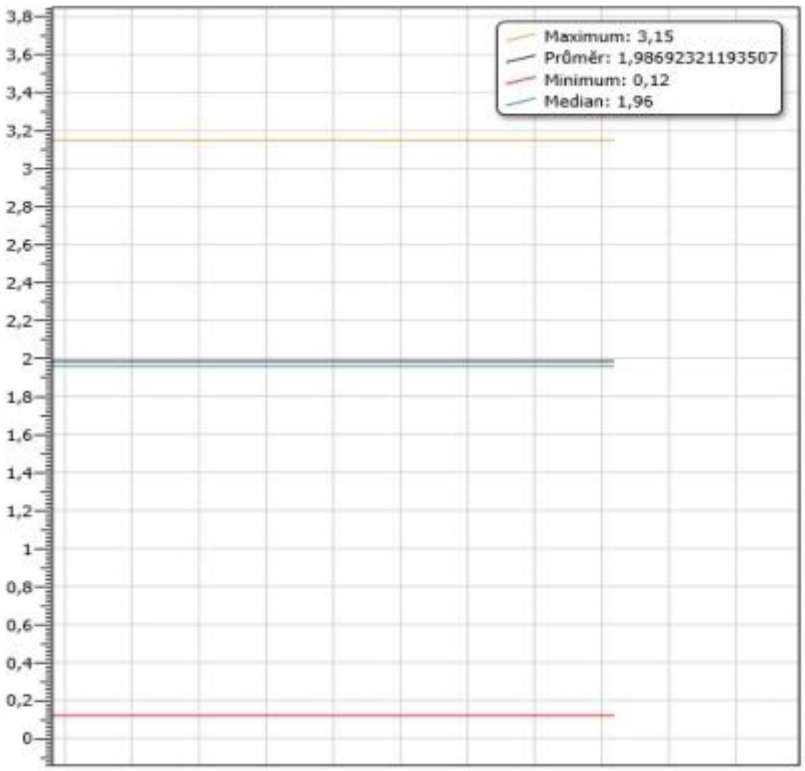


Obrázek 5.2: Výpis obsahu testovací zdravotnické databáze s ovládacími prvky softwaru

5.3 Realizace základních funkcí

Realizace statistických funkcí je v jazyku C# o něco složitější záležitost, než ve vývojovém prostředí Matlab. Vývojové prostředí Matlab je přímo matematický software, který má již statistické funkce předdefinované. V programovacím jazyku C# je tomu však jinak. Zde je vše závislé na vývojových knihovnách a funkcích v nich obsažených.
Funkce, jako minumum, maximum a aritmetický průměr jsou zde k dispozici. Ovšem funkce medián musela být naprogramována. Tato funkce byla naprogramována tak, že při lichém počtu výběrového datoveho souboru pro analýzu vybere prostřední hodnotu z datového souboru, což odpovídá mediánu.

Při sudém počtu výběrového datového souboru pro analýzu vybere ze souboru dat vždy dvě prostřední hodnoty, které zprůměruje, což také odpovídá hodnotě mediánu. Nutno podotknout, že takový výpočet mediánu vyžaduje aplikaci daných funkcí na seřazený datový soubor.



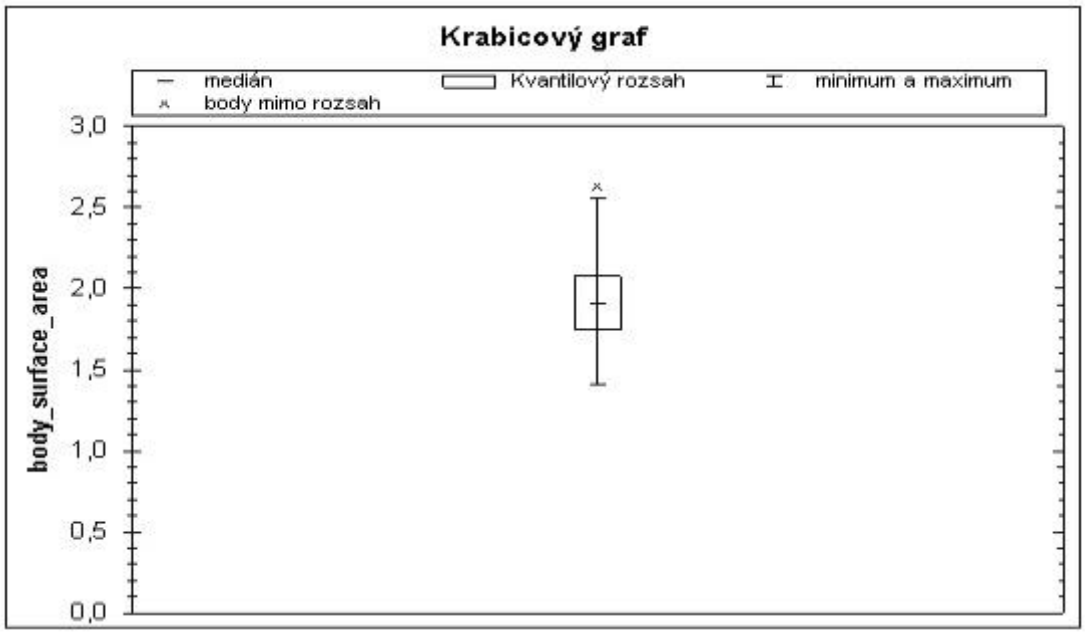
Obrázek 5.3: Grafické zobrazení základních statistických funkcí

5.4 Realizace grafů

Grafy jsou realizovány pomocí externích rozšíření programu. Bylo tedy využito externích knihoven „Zedgraph“ a „Dynamic Data Display“. Knihovna „Zedgraph“ nabízí širší možnosti zobrazování grafů, proto je tato knihovna využívána pro zobrazení krabicového grafu. Knihovna „Dynamic Data Display“ naopak vyniká v zobrazování dat v reálném čase. Tato knihovna také vyniká v lepším grafickém zobrazování dat. Proto byla tato knihovna použita pro zobrazení základních statistických funkcí programu. Nejedná se tedy o graf, ale pouze o grafické zobrazení hodnot, které je po vizuální stránce pro uživatele programu přijatelnější.

5.4.1 Krabicový graf

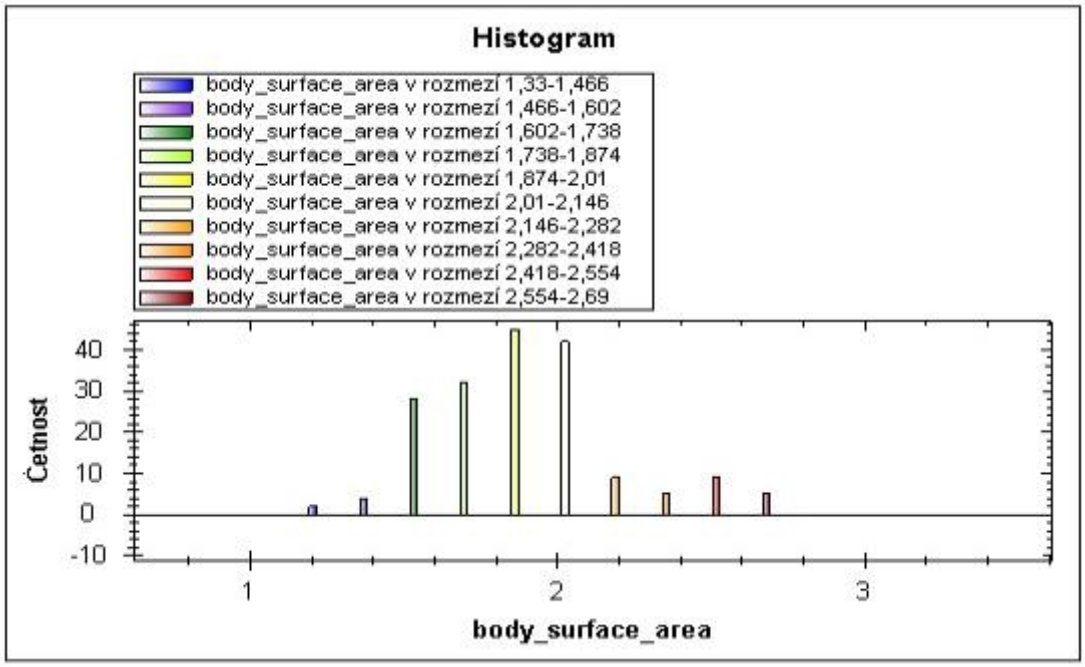
Pro zobrazení a vykreslení krabicového grafu aplikace využívá knihovnu „Zedgraph“. Jelikož tato knihovna neumí přímo zobrazit krabicový graf z dat, které má k dispozici, musel být tento typ grafu takzvaně sestrojen. Vykreslení grafu předchází výpočet prvního a třetího kvantilu ze souboru dat. Dále výpočet minima, maxima, mediánu a odlehlých hodnot. Výsledný krabicový graf má poté podobu, jakou lze vidět na následujícím obrázku.



Obrázek 5.4: Krabicový graf s vybočující hodnotou z analyzovaného souboru

5.4.2 Histogram

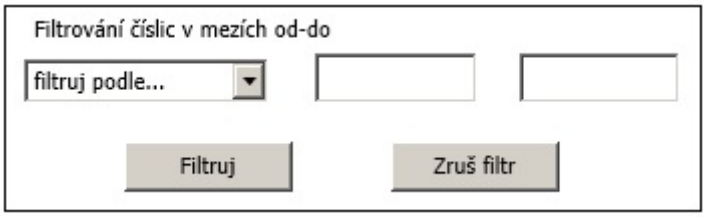
Pro vykreslování histogramu je využito stejné knihovny jako pro vykreslování krabicového grafu. Pro přípravu dat k vykreslení histogramu je využito matematické knihovny „MathNet“. Tato knihovna rozdělí hodnoty vstupních dat na deset intervalů. Každému intervalu je přiřazena četnost, která odpovídá počtu hodnot, které spadají do daného intervalu. Výslednou podobu histogramu lze vidět na následujícím obrázku.



Obrázek 5.5: Histogram vykreslený statistickým software

5.5 Třídění číslicových dat

Je důležité, aby uživatel měl k dispozici možnost třídění dat z databáze. Tento prvek je užitečný jak při hledání dat z databáze, tak při hledání pacientů z databáze, kteří patří do určité skupiny pacientů. Třídění v softwaru zajišťují prvky roletkového menu s dvěma vstupními poli pro zadání rozsahu hledaných hodnot z vybrané tabulky v databázi. Bude-li uživatel chtít zobrazit pouze takové záznamy z databáze, které odpovídají pouze jednomu parametru (nikoliv tedy rozsahu parametrů), zadá uživatel tento paremetr do obou polí rozsahu. Program poté vypíše pouze takové záznamy, které odpovídají právě této jedné zadané hodnotě.



Obrázek 5.6: Filtr číslicových dat

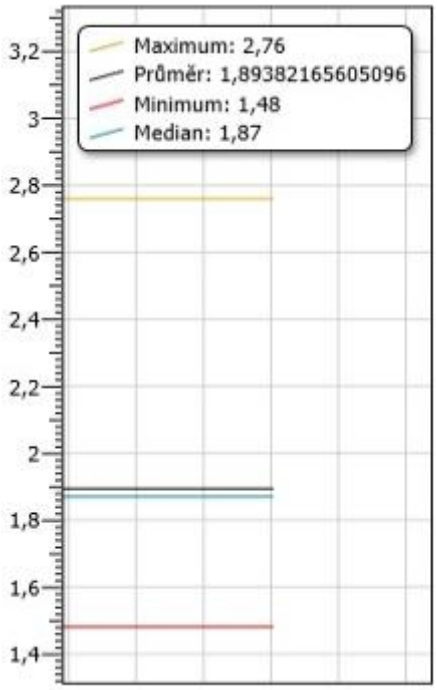
Kapitola 6
Testování softwaru a výsledků analýzy

Pro testování softwaru a výsledků jeho analýzy byla použita testovací zdravotnická databáze s pravdivými údaji o pacientech a výsledcích z měření jednotlivých testů. Tato zdravotnická databáze byla poskytnuta vedoucím bakalářské práce.

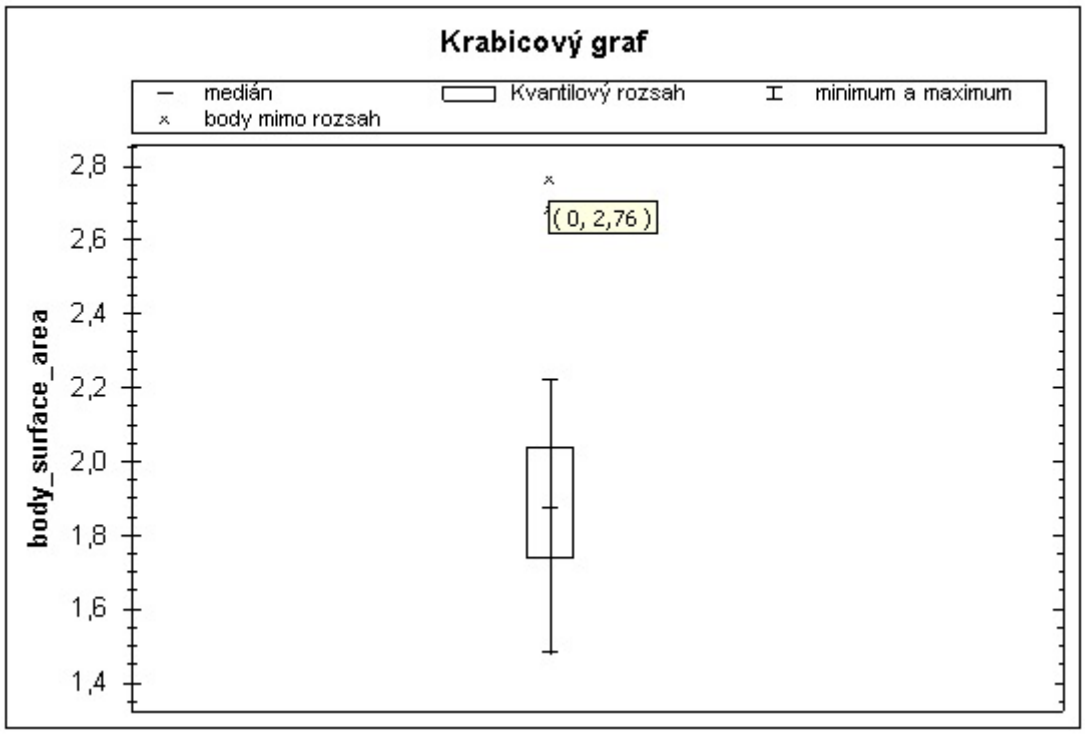
Pro testování navrženého softwaru byly použity naměřená data „boady surface area“. Tento naměřený údaj vypovídá o naměřené ploše těla jedince. Velikost tohoto údaje závisí na jeho stavbě těla. Čím je tento údaj vyšší, tím má jedinec větší plochu těla. Vysoká hodnota povrchu těla, může být známkou obezity vyšetřované osoby. Plocha těla je také údaj závislý na věku vyšetřovaného. Jinou plochu těla bude mít kojenec a jinou plochu těla bude mít zase dospělý člověk. Proto pro testování softwaru byli vybráni pacienti, kteří jsou ve stejné věkové kategorii. Konkrétně byli vybráni pacienti ve věku dvaceti, třiceti a čtyřiceti let. U těchto tří kategorií pacientů bude tedy sledována naměřená hodnota „boady surface area“.

6.1 Výsledky analýzy programu

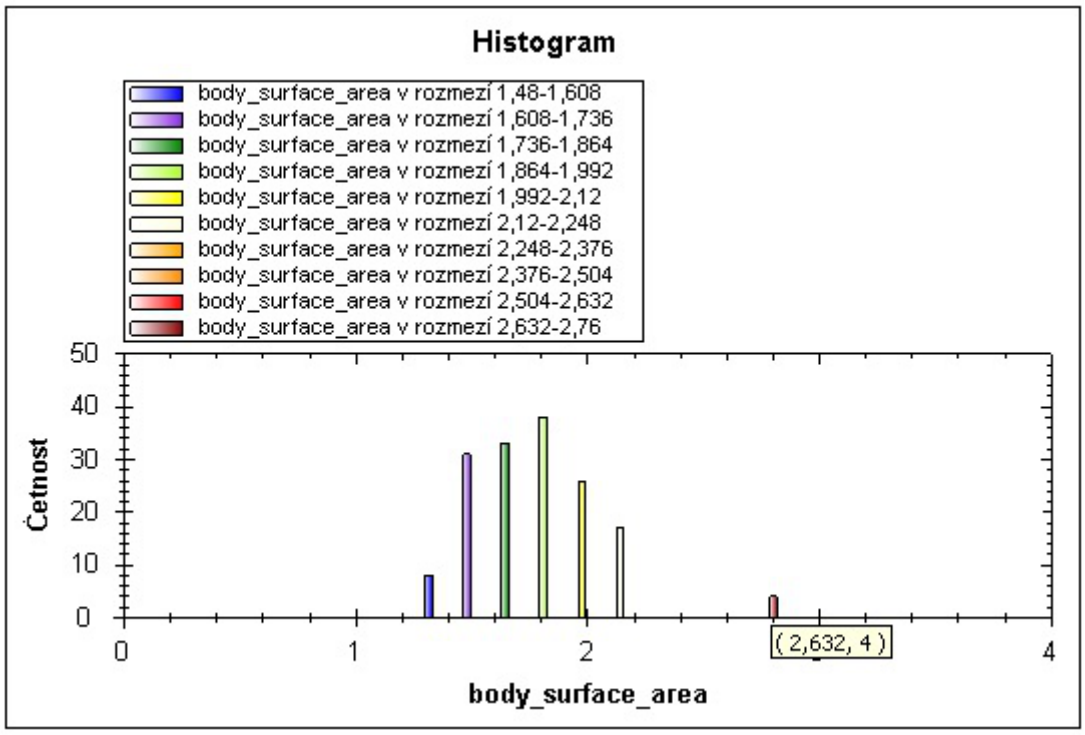
6.1.1 Pacienti ve věku dvaceti let



Obrázek 6.1: Analýza hodnoty „boady surface area“ pacientů ve věku dvaceti let

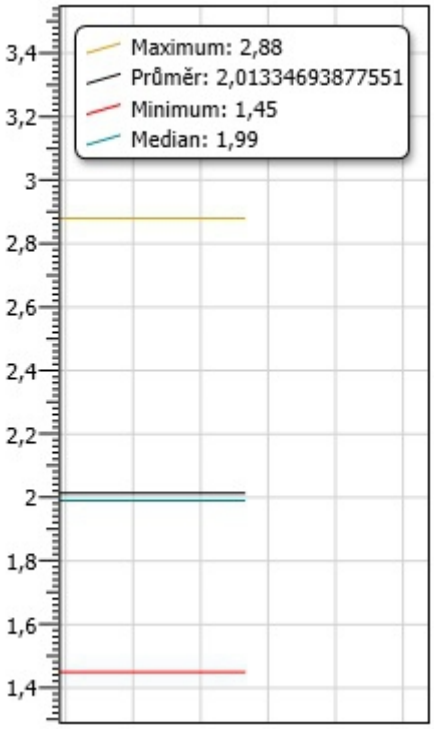


Obrázek 6.2: Krabicový graf pacientů ve věku dvaceti let z hodnoty „boady surface area“

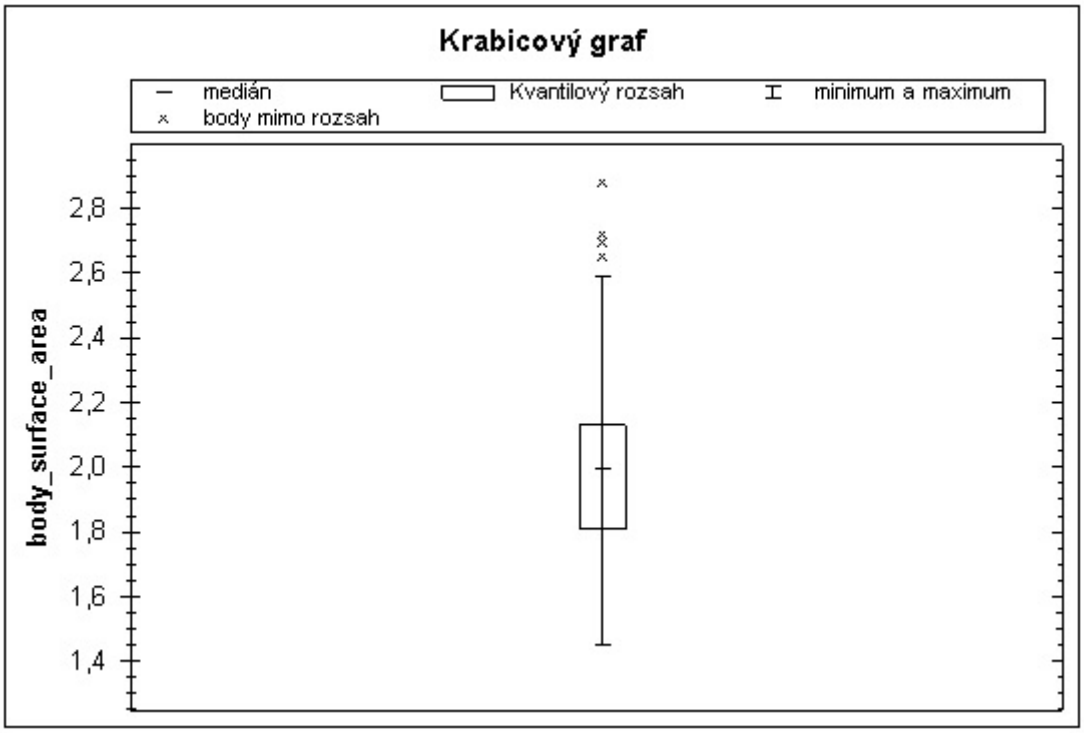


Obrázek 6.3: Histogram pacientů ve věku dvaceti let z hodnoty „boady surface area“

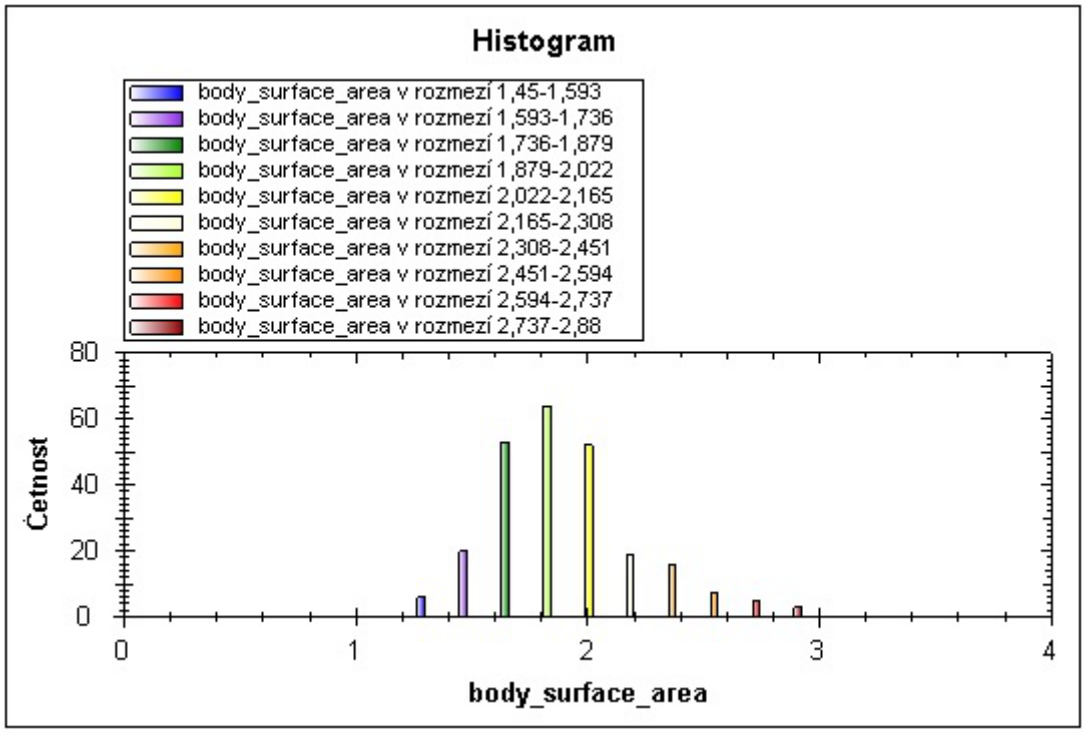
6.1.2 Pacienti ve věku třiceti let



Obrázek 6.4: Analýza hodnoty „boady surface area“ pacientů ve věku třiceti let

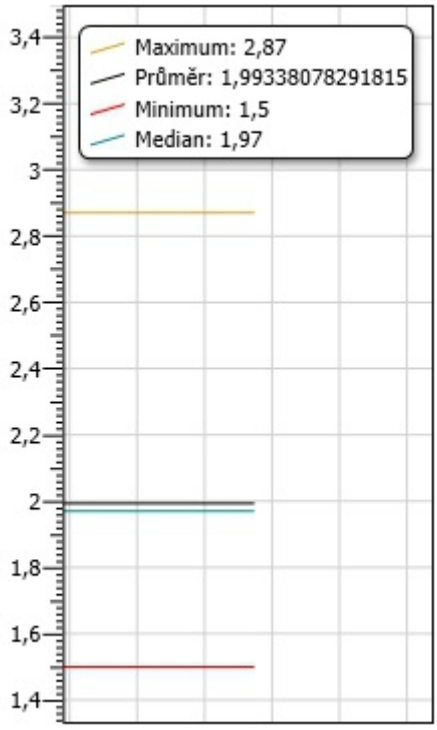


Obrázek 6.5: Krabicový graf pacientů ve věku třiceti let z hodnoty „boady surface area“

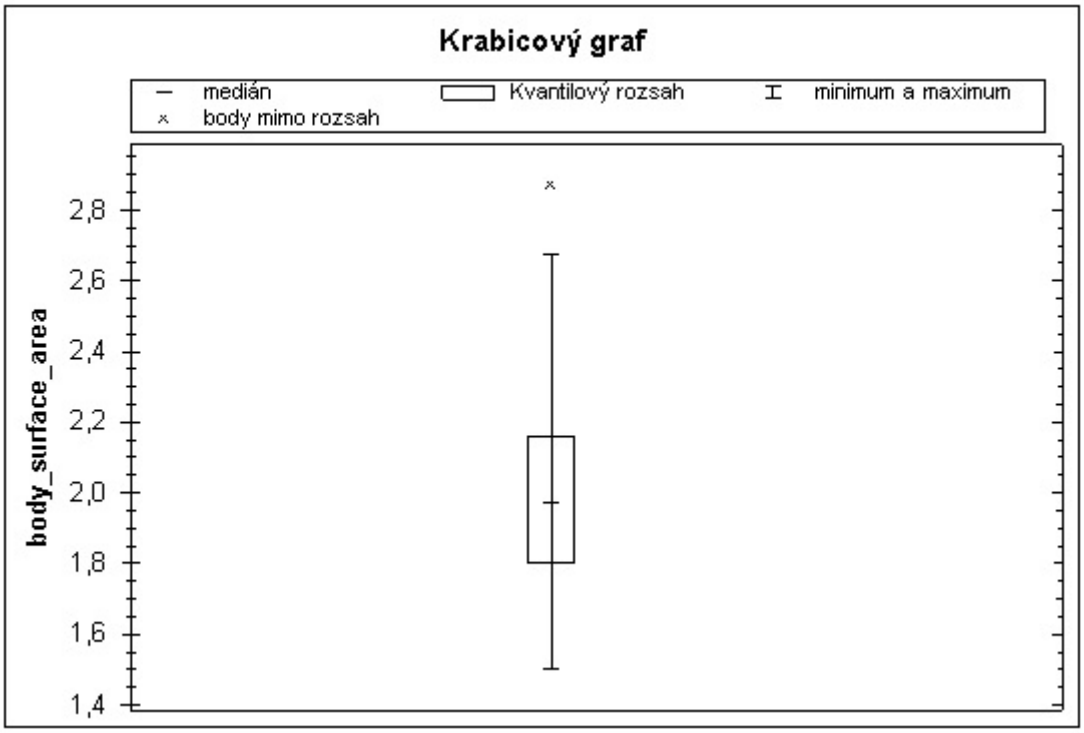


Obrázek 6.6: Histogram pacientů ve věku třiceti let z hodnoty „boady surface area“

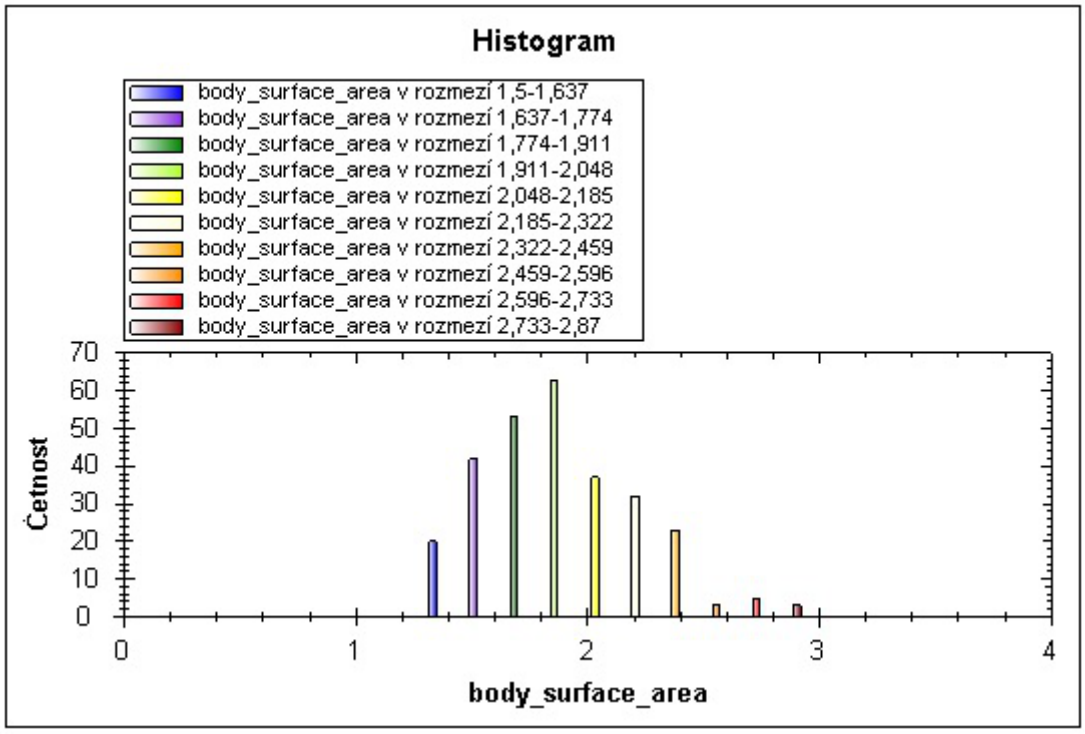
6.1.3 Pacienti ve věku čtyřiceti let



Obrázek 6.7: Analýza hodnoty „boady surface area“ pacientů ve věku čtyřiceti let



Obrázek 6.8: Krabicový graf pacientů ve věku čtyřiceti let z hodnoty „boady surface area“



Obrázek 6.9: Histogram pacientů ve věku čtyřiceti let z hodnoty „boady surface area“

6.2 Zhodnocení výsledků analýzy

Na testovaných skupinách pacientů lze vidět, že průměrná hodnota plochy těla s věkem roste, ale ve věku čtyřiceti let nepatrně klesá. Tento jev může být způsoben svrašťěním kůže ve vyšším věku. Dále jde na krabicovém grafu vidět, že v každé skupině pacientů jsou pacienti, kteří svými hodnotami vybočují od ostatních. Tito pacienti, kteří vybočují jsou pacienti, kteří jsou obézní, mají tedy zvýšenou plochu těla tím, že jejích kůže je více natažená. Na histogramech lze vidět rozložení intervalů a jejich četností z analyzovaného souboru dat.

Kapitola 7
Závěr

7.1 Zhodnocení výsledků práce

V první kapitole této práce byly popsány některé metody energometrických měření. Jedná se o měření, kterými se zjišťuje množství vydaných kalorií tělem pacienta při různých činnostech. V kapitole druhé byly popsány některé metody statistického zpracování dat a jejich využití. O některé z těchto metod statistického vyhodnocení dat se opírá vyvíjený statistický software. Třetí teoretická kapitola pojednává o relačních databázích a jejich obsluhování prostřednictvím databázového jazyka SQL. Kapitola čtvrtá pojednává o některých postupech při vývoji softwaru.
Cílem této bakalářské práce bylo vyvinout statistický software v některém z programovacích jazyků. Z hlediska budoucnosti softwaru a požadavcích na jeho dalsí vylepšování jsem si zvolil programovací jazyk C#. V tomto programovacím jazyku byly za pomocí volně dostupných knihoven „Zedgraph“, „Dynamic Data Display“ a statistické knihovny „MathNet“ zkonstruovány statistickké grafy jako histogram a krabicový graf. Do softwaru bylo přidáno také třídění číslicových dat ve zobrazené databázové tabulce podle všech sloupců v ní obsažených, což uživateli usnadní výběr specifických dat pro analýzu.

7.2 Praktické uplatnění práce

Vyvinutý software je primárně určen a optimalizován pro statistické vyhodnocování naměřených dat ze zdravotnické databáze. Umožňuje lékařům zobrazit statistické grafy z nichž mohou posoudit, zda se ve vybraném datovém souboru nachází pacient, který svou hodnotu vybočuje od hodnot přípustných. Je-li tomu tak, je už na lékaři posoudit závažnost této odchylky.
Software je psán univerzálně, což v praxi znamená, že se může napojit na jakoukoliv databázi typu MySQL. Software si tedy poradí i s databází nezdravotnického typu.

7.3 Doporučení pro další postup

Navrhnutý software může být rozšířen o jakoukoliv statistickou analýzu dat, která bude přínosem pro lékaře při posuzování statistických závislostí. Jde zejména o rozšíření grafického zobrazování výsledků analyzovaných dat. Také jako velmi užitečné rozšížení bych viděl v rozšíření softwaru o pokročilé statistické analýzy dat jako je například analýza hlavních komponent nebo faktorová analýza. Další postupy rozšiřování programu jsou vysoce závislé na požadavcích na software ze strany lékařských výzkumných pracovníků, pro které je tento software primárně určen.

Literatura

[1] JIRÁK, Zdeněk. *Fyziologie pro bakalářské studium na ZSF OU*. 2., přeprac. vyd. Ostrava: Ostravská univerzita v Ostravě, Zdravotně sociální fakulta, 2007, 249 s. ISBN 978-80-7368-234-7.

[2] PETR, Miroslav. *Energometrie a kalorimetrie* [online]. [cit. 2013-01-21]. Dostupné z: http://www.florbalovytrener.cz/wp-content/uploads/2012/04/Energometrie_a_kalorimetrie.pdf.

[3] SIGMUND, Erik. *Vybrané metodologické aspekty etiky výzkumu* [online]. Univerzita Palackého v Olomouci, fakulta tělesné kultury, 2012 [cit. 2013-01-21]. Dostupné z: http://ftk.upol.cz/fileadmin/user_upload/FTK-dokumenty/Komise/Metodologicke_aspekty_etiky_vyzkumu.pdf

[4] HARRIS, James Arthur; BENEDICT, Francis Gano. *A biometric study of human basal metabolism* [online]. *Proc Natl Acad Sci U S A*. Washington : NAS, 1918 [cit. 2013-01-21]. Dostupné z: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1091498/?tool=pmcentrez

[5] HARRIS, James Arthur; BENEDICT, Francis Gano. *A biometric study of human basal metabolism in man* [online]. *Proc Natl Acad Sci U S A*. Washington : NAS, 1918 [cit. 2013-01-21]. Dostupné z: http://booksnow2.scholarsportal.info/ebooks/oca1/12/biometricstudyof00harruoft//biometricstudyof00harruoft.pdf

[6] HÁJOVSKÝ, Radovan. *Měření a zpracování dat pro obor biomedicínská technika*. 1. vyd. Ostrava: VŠB - Technická univerzita Ostrava, 2007, 101 s. ISBN 978-80-248-1588-6.

[7] DOHNAL, Luděk. *Analysa rozptylu – ANOVA* [online]. [cit. 2013-01-21]. Dostupné z: http://www1.lf1.cuni.cz/~ldohna/publik/Kap_7_ANOVA.pdf

[8] PRAUZEK, Michal. *Klasifikace typologie metabolismu na základě analýzy dat z energometrických testů*. Ostrava, 2011. Disertační práce. Technická Univerzita Ostrava, Fakulta Elektrotechniky a Informatiky, katedra Měřící a Řídící techniky.

[9] POPELKA, Ondřej. *Aplikační programové vybavení* [online].2013 [cit. 2013-03-10]. Dostupné z: https://akela.mendelu.cz/~xpopelka/cs/apv/

[10] *Vodopádový model - Wikipedie* [online]. 2013 [cit. 2013-03-11]. Dostupné z: http://cs.wikipedia.org/wiki/Vodop%C3%A1dov%C3%BD_model

[11] *File: Software Development Spiral.svg - Wikimedia Commons* [online]. 2009 [cit. 2013-04-28]. Dostupné z: http://commons.wikimedia.org/wiki/File:Software_Development_Spiral.svg

Příloha A

Strojový výpis hlavního programu

```
using System;
using System.Collections;
using System.Collections.ObjectModel;
using System.Collections.Generic;
using System.Configuration;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Data;
using System.Data.OleDb;
using MySql.Data;
using MySql.Data.MySqlClient;
using System.Drawing;
using System.IO;
using Microsoft.Research.DynamicDataDisplay; // Core functionality
using Microsoft.Research.DynamicDataDisplay.DataSources; // EnumerableDataSource
using Microsoft.Research.DynamicDataDisplay.PointMarkers; // CirclePointMarker
using MathNet.Numerics.Distributions;
using MathNet.Numerics.Signals;
using MathNet.Numerics.Statistics;


namespace Database
{
       /// <summary>
       /// Interaction logic for MainWindow.xaml
       /// </summary>
    public partial class MainWindow : Window
    {
        //ArrayList graphValues = new ArrayList();
        List<int> graphValues = new List<int>();
        int plotGraphClick = 0;
        string id = string.Empty;

        public MainWindow()
        {
            InitializeComponent();
        }
```

```
private void btn_tbTest_Click(object sender, RoutedEventArgs e)
        {
            MySqlConnection conn = new MySqlConnection(ConnStr());
            try
            {
                conn.Open();
                string sql = "SHOW TABLES";
                MySqlCommand cmd = new MySqlCommand(sql, conn);
                MySqlDataReader result = cmd.ExecuteReader();
                while (result.Read())
                {
                    textBox1.AppendText(result.GetString(0) + "\n");
                }
                result.Close();
                conn.Close();
            }
            catch (MySqlException)
            {
                System.Windows.MessageBox.Show("Připojení k databázi selhalo!" + "\n"
+ "Pro připojení zadejte správné údaje.");
            }
        }

        private void btn_pripojit_Click(object sender, RoutedEventArgs e)
        {
            MySqlConnection conn = new MySqlConnection(ConnStr());
            try
            {
                conn.Open();
            }
            catch (MySqlException)
            {
                System.Windows.MessageBox.Show("Zadejte správné údaje pro připojení do
databáze.", "Připojení selhalo!");
            }
            finally
            {
                if (conn.State.ToString().Equals("Open"))
                {
                    System.Windows.MessageBox.Show("Vše v pořádku, nyní můžete
vyhodnocovat data.", "Databáze připojena.");
                }
            }
        }

        private string ConnStr()
        {
            string ip;
            string port;
            string databaze;
            string jmeno;
            string heslo;

            ip = "server=" + tb_ip.Text + ";";
            port = "port=" + tb_port.Text + ";";
            databaze = "database=" + tb_databaze.Text + ";";
            jmeno = "user=" + tb_login.Text + ";";
            heslo = "password=" + tb_heslo.Text + ";";
```

```
            return ip + jmeno + databaze + port + heslo;
        }

        private void lbRefresh()
        {
            ArrayList itemsList = new ArrayList();
            int count = dataGrid1.Columns.Count;
            //string[] columnHeader = new string[count];

            for (int i = 0; i < count; i++)
            {
                itemsList.Add(dataGrid1.Columns[i].Header.ToString());
            }

            cb_columnSelect.ItemsSource = itemsList;
        }

        private void cbRefresh()
        {
            ArrayList itemsList = new ArrayList();
            int count = dataGrid1.Columns.Count;

            for (int i = 0; i < count; i++)
            {
                itemsList.Add(dataGrid1.Columns[i].Header.ToString());
            }

            filterList.ItemsSource = itemsList;
        }

        public void zgbutton_Click(object sender, RoutedEventArgs e)
        {
            if (plotGraphClick > 0)
            { dddchart_1.Children.RemoveAll(typeof(LineGraph)); }
            else
            { plotGraphClick++; }

            List<string> selectedValueStringList = new
List<string>(dataGrid1.SelectedItems.Count); //create list
            double[] selectedValueArray = new double[dataGrid1.SelectedItems.Count];
//hodnoty vybrané do grafu
            string id = string.Empty; //define string

            if (dataGrid1.SelectedItems.Count > 0)    //when selection applied..
            {
                for (int i = 0; i < dataGrid1.SelectedItems.Count; i++) //go row by
row in selected column above
                {
                    try
                    {
                        System.Data.DataRowView selectedFile =
(System.Data.DataRowView)dataGrid1.SelectedItems[i];
                        string str =
Convert.ToString(selectedFile.Row.ItemArray[cb_columnSelect.SelectedIndex]);
                        selectedValueStringList.Add(str);
                    }
                    catch (Exception ex)
                    { System.Windows.MessageBox.Show(ex.Message, "error"); }
                }
            }
            string histLabelX = cb_columnSelect.SelectedItem.ToString();
            plotBasicStatistic(selectedValueStringList, selectedValueArray);
```

```
            showBoxplot(selectedValueStringList, selectedValueArray, histLabelX);
            histPlot(selectedValueArray, histLabelX);
        }

        public void showBoxplot(List<string> stringValues, double[] valuesToWhisker,
string histLabelX)
        {
            for (int i = 0; i < stringValues.Count; i++) //parse from string to double
array
            {
                valuesToWhisker[i] = double.Parse(stringValues[i]);
            }
            Array.Sort(valuesToWhisker);

            ZedGraphTest.MainWindow boxplot = new
ZedGraphTest.MainWindow(valuesToWhisker, histLabelX);

            boxplot.Show();
        }

        public void plotBasicStatistic(List<string> stringValues, double[]
doubleValues)
        {
            for (int i = 0; i < stringValues.Count; i++) //parse from string to double
array
            {
                doubleValues[i] = double.Parse(stringValues[i]);
            }
            Array.Sort(doubleValues);

            //--------minimum--------------------
            double[] minPlot = CreateAxesCycle(doubleValues.Min(), 0);
            //--------maximum----------
            double[] maxPlot = CreateAxesCycle(doubleValues.Max(), 0);
            //--------average--------------
            double[] averagePlot = CreateAxesCycle(doubleValues.Average(), 0);
            //--------horizontal axis-------
            double[] xValue = CreateAxesCycle(double.NaN, 1);
            //--------median---------------
            double median = Median(doubleValues);
            double[] medianPlot = CreateAxesCycle(median, 0);
            //------------------------

            CompositeAndAddGrapg(xValue, maxPlot,
System.Windows.Media.Colors.Goldenrod, 1, "Maximum: " +
doubleValues.Max().ToString());
            CompositeAndAddGrapg(xValue, averagePlot,
System.Windows.Media.Colors.Black, 1, "Průměr: " + doubleValues.Average().ToString());
            CompositeAndAddGrapg(xValue, minPlot, System.Windows.Media.Colors.Red, 1,
"Minimum: " + doubleValues.Min().ToString());
            CompositeAndAddGrapg(xValue, medianPlot,
System.Windows.Media.Colors.DarkCyan, 1, "Median: " + median.ToString());

            dddchart_1.HorizontalAxis.Remove();
            dddchart_1.FitToView();
        }
```

```
private double Median(double[] sortedDataArray)
        {
            if (sortedDataArray.Count() % 2 == 0)
            {
                // It's even
                int arrayNumber = sortedDataArray.Count() / 2;
                double dataReturn = (sortedDataArray[arrayNumber] +
sortedDataArray[arrayNumber - 1]) / 2;
                return dataReturn;
            }
            else
            {
                // It's odd
                int arrayNumber = ((sortedDataArray.Count() - 1) / 2);
                return sortedDataArray[arrayNumber];
            }
        }

        private void CompositeAndAddGrapg(double[] xData, double[] yData,
System.Windows.Media.Color color, double lineThickness, string legend)
        {
            var xDataSource = xData.AsXDataSource();
            var yDataSource1 = yData.AsYDataSource();
            CompositeDataSource graphDataSource = xDataSource.Join(yDataSource1);
            dddchart_1.AddLineGraph(graphDataSource, color, lineThickness, legend);
        }

        private double[] CreateAxesCycle(double valueToPlot, int isHorizontalAxis)
        {
            double[] data = new double[2];

            if (isHorizontalAxis != 0)
            {
                for (int i = 0; i < 2; i++)
                {
                    data[i] = i;
                }
            }
            else
            {
                for (int i = 0; i < 2; i++)
                {
                    data[i] = valueToPlot;
                }
            }
            return data;
        }

        private void btn_gridFill_Click(object sender, RoutedEventArgs e)
        {
            sql_gridRoutine("SHOW TABLES");
        }
```

```
public void sql_gridRoutine(string sqlCmd)
        {
            MySqlConnection conn = new MySqlConnection(ConnStr());
            try
            {
                conn.Open();
                //string sql = "SHOW TABLES";
                using (MySqlCommand cmd = new MySqlCommand(sqlCmd, conn))
                {
                    DataTable dt = new DataTable();
                    MySqlDataAdapter da = new MySqlDataAdapter(cmd);
                    da.Fill(dt);
                    //da.Update(dt);
                    dataGrid1.ItemsSource = dt.DefaultView;
                }
                conn.Close();
            }
            catch (Exception)
            { System.Windows.MessageBox.Show("Zkontrolujte filtrační meze." + "\nPro
desetinnou čárku používejte znak '.'", "Filtrování hodnot selhalo!"); }
        }

        private void btn_intoDB_Click(object sender, RoutedEventArgs e)
        {
            if (dataGrid1.SelectedIndex != -1)
            {
                DataGridColumn dataGridCol = dataGrid1.Columns[0]; //sloupec id
                id = ((TextBlock)dataGridCol.GetCellContent
                    (dataGrid1.SelectedItem)).Text;
                sql_gridRoutine("SELECT * FROM " + id);
            }
            lbRefresh();
            cbRefresh();
        }

        private void graphPlot_Click(object sender, RoutedEventArgs e)
        {
            int length = 500;
            double[] data = new double[length];
            Random randGen = new Random();
            for (int i = 0; i < length; ++i)
            {
                data[i] = randGen.NextDouble();
            }

            var hist = new MathNet.Numerics.Statistics.Histogram(data, 10, 0, 1);
            var bucket3count = hist[0].Count;

        }
```

```
 public void histPlot(double[] selectedValueArray, string histLabelX)
        {
            Array.Sort(selectedValueArray);
            int minimum = Convert.ToInt32(selectedValueArray.Min());
            int maximum = Convert.ToInt32(selectedValueArray.Max());
            IEnumerable<double> data = selectedValueArray;
            var histPlotBuckets = new MathNet.Numerics.Statistics.Histogram(data, 10);

            Collection<double[]> ColHistPoints = new Collection<double[]>();
            histpointsSetup(ColHistPoints, histPlotBuckets);
            ZedgraphHistplot.MainWindow histplot = new
ZedgraphHistplot.MainWindow(ColHistPoints, histLabelX);
            histplot.Show();
        }

        private void histpointsSetup(Collection<double[]> CollectionForPoints,
Histogram buckets)
        {
            for (int i = 0; i < buckets.BucketCount; i++)
            {
                var vyska1 = buckets[i].Count;
                var lower1 = buckets[i].LowerBound;
                double[] X = new double[1] { lower1 };
                double[] Y = new double[1] { vyska1 };
                CollectionForPoints.AddMany(X, Y);
            }

            var last = buckets[9].UpperBound;
            double[] lastRange = new double[1] { last };
            CollectionForPoints.Add(lastRange);
        }

        private void btn_filter_Click(object sender, RoutedEventArgs e)
        {
            string A = tb_od.Text.Replace(".", ",");
            string B = tb_do.Text.Replace(".", ",");
            if (double.Parse(A) > double.Parse(B))
            { System.Windows.MessageBox.Show("Meze jsou zadány opačně.", "Filtrování
hodnot selhalo!"); }
            else if (double.Parse(A) == double.Parse(B))
            {
                int x = dataGrid1.Items.Count; //spočítej, vypsané záznamy
                sql_gridRoutine("SELECT * FROM " + id + " WHERE " +
filterList.SelectedItem.ToString() + "=" + tb_od.Text);
                int y = dataGrid1.Items.Count; // spočítej, záznamy po filtraci
                if (x != y) // byl filtr aplikovan ?
                { System.Windows.MessageBox.Show("Filtr byl aplikován.", "Filtrování
hodnot proběhlo."); }
            }
            else if (double.Parse(A) < double.Parse(B))
            {
                int x = dataGrid1.Items.Count; //spočítej, vypsané záznamy
                sql_gridRoutine("SELECT * FROM " + id + " WHERE " +
filterList.SelectedItem.ToString() + ">=" + (tb_od.Text) + " AND " +
filterList.SelectedItem.ToString() + "<=" + (tb_do.Text));
                int y = dataGrid1.Items.Count; // spočítej, záznamy po filtraci
                if (x != y) // byl filtr aplikovan ?
                { System.Windows.MessageBox.Show("Filtr byl aplikován.", "Filtrování
hodnot proběhlo."); }
```

```
            }
        }


        private void btn_filterClear_Click(object sender, RoutedEventArgs e)
        {
            sql_gridRoutine("SELECT * FROM " + id);
        }

        private void btn_tbTest_Click_1(object sender, RoutedEventArgs e)
        {

        }
    }
}
```

Příloha B

Strojový výpis vykreslování krabicového grafu

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ZedGraph;
using System.Windows.Forms.DataVisualization.Charting;


namespace ZedGraphTest
{
    public partial class ZedGraphUserControl : UserControl
    {
        double[] med = new double[3];

        public ZedGraphUserControl(double[] DataToWhisker, string histLabelX)
        {
            InitializeComponent();
            CreateGraph(zgcGraph, DataToWhisker, histLabelX);
            SetSize();
        }
        public new int Width
        {
            get { return zgcGraph.Width; }
            set { zgcGraph.Width = value; }
        }
        public new int Height
        {
            get { return zgcGraph.Height; }
            set { zgcGraph.Height = value; }
        }
```

```
 public void CreateGraph(ZedGraphControl zgc, double[] DataToWhisker, string
histLabelX)
        {

            List<double[]> dataList = new List<double[]>();
            List<string> names = new List<string>();
            dataList.Add(DataToWhisker);

            names.Add("Kvantilový rozsah");



            med[0] = Median(DataToWhisker);

            GraphPane myPane = zgc.GraphPane;

            myPane.BarSettings.Type = BarType.Overlay;
            myPane.XAxis.IsVisible = false;
            myPane.Legend.IsVisible = true;
            myPane.Title.Text = "Krabicový graf";
            myPane.YAxis.Title.Text = histLabelX;


            BoxPlot(dataList, names, zgc);

            myPane.AxisChange();
        }
        public void BoxPlot(List<double[]> data, List<string> names, ZedGraphControl
zgc)
        {
            GraphPane myPane = zgc.GraphPane;
            for (int i = 0; i < data.Count; i++)
            {
                //median of each array
                PointPairList medians = new PointPairList();
                //75th and 25th percentile, defines the box
                PointPairList hiLowList = new PointPairList();
                //+/- 1.5*Interquartile range, extentent of wiskers
                PointPairList barList = new PointPairList();
                //outliers
                PointPairList outs = new PointPairList();
                //Add the values

                medians.Add(i, med[i]);
                hiLowList.Add(i, percentile(data[i], 75), percentile(data[i], 25));
                double iqr = 1.5 * (percentile(data[i], 75) - percentile(data[i],
25));
                double upperLimit = percentile(data[i], 75) + iqr;
                double lowerLimit = percentile(data[i], 25) - iqr;
                //The wiskers must end on an actual data point
                barList.Add(i, ValueNearestButGreater(data[i], lowerLimit),
ValueNearestButLess(data[i], upperLimit));
                //Sort out the outliers
```

```
            foreach (double aValue in data[i])
               {
                   if (aValue > upperLimit)
                   {
                       outs.Add(i, aValue);
                   }
                   if (aValue < lowerLimit)
                   {
                       outs.Add(i, aValue);
                   }
               }
               //Plot the items, first the median values
               CurveItem meadian = myPane.AddCurve("medián", medians, Color.Black,
SymbolType.HDash);

               LineItem myLine = (LineItem)meadian;
               myLine.Line.IsVisible = false;
               myLine.Symbol.Fill.Type = FillType.Solid;
               //Box
               HiLowBarItem myCurve = myPane.AddHiLowBar(names[i], hiLowList,
Color.Black);
               myCurve.Bar.Fill.Type = FillType.None;
               //Wiskers
               ErrorBarItem myerror = myPane.AddErrorBar("minimum a maximum",
barList, Color.Black);
               //Outliers
               CurveItem upper = myPane.AddCurve("body mimo rozsah", outs,
Color.Black, SymbolType.XCross);
               LineItem bLine = (LineItem)upper;
               bLine.Symbol.Size = 3;
               bLine.Line.IsVisible = false;

           }
       }
       private double ValueNearestButLess(double[] data, double number)
       {
           double lowNums = double.MinValue;
           foreach (double n in data)
           {
               if (n <= number)
               {
                   lowNums = Math.Max(n, lowNums);
               }
           }
           return lowNums;
       }
       private double ValueNearestButGreater(double[] data, double number)
       {
           double lowNums = double.MaxValue;
           foreach (double n in data)
           {
               if (n >= number)
               {
                   lowNums = Math.Min(n, lowNums);
               }
           }
           return lowNums;
       }
```

```
        private double percentile(double[] Data, double p)
         {
             //Percentile lowp = new Percentile(Data);
             //lowp.Method = PercentileMethod.Excel;
             //
[url]http://www.codeproject.com/KB/recipes/DescriptiveStatisticClass.aspx[/url]
             Array.Sort(Data);
             if (p >= 100.0d) return Data[Data.Length - 1];
             double position = (double)(Data.Length + 1) * p / 100.0;
             double leftNumber = 0.0d, rightNumber = 0.0d;
             double n = p / 100.0d * (Data.Length - 1) + 1.0d;
             if (position >= 1)
             {
                 leftNumber = Data[(int)System.Math.Floor(n) - 1];
                 rightNumber = Data[(int)System.Math.Floor(n)];
             }
             else
             {
                 leftNumber = Data[0]; // first data
                 rightNumber = Data[1]; // first data
             }
             if (leftNumber == rightNumber)
                 return leftNumber;
             else
             {
                 double part = n - System.Math.Floor(n);
                 return leftNumber + part * (rightNumber - leftNumber);
             }
         }
         private void SetSize()
         {
             zgcGraph.Location = new System.Drawing.Point(10, 10);
             // Leave a small margin around the outside of the control
             zgcGraph.Size = new System.Drawing.Size((int)this.Width - 20,
                                 (int)this.Height - 20);
         }

         private double Median(double[] sortedDataArray)
         {
             //int Num = 5;

             if (sortedDataArray.Count() % 2 == 0)
             {
                 // It's even
                 int arrayNumber = sortedDataArray.Count() / 2;
                 double dataReturn = (sortedDataArray[arrayNumber] +
sortedDataArray[arrayNumber - 1]) / 2;
                 return dataReturn;
             }
             else
             {
                 // It's odd
                 int arrayNumber = ((sortedDataArray.Count() - 1) / 2);
                 return sortedDataArray[arrayNumber];
             }

         }
     }


}
```

Příloha C

Strojový výpis vykreslování histogramu

```
using System;
using System.Collections;
using System.Collections.ObjectModel;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ZedGraph;


namespace ZedgraphHistplot
{
    public partial class zgucHistplot : UserControl
    {
        double[] med = new double[3];

        public zgucHistplot(Collection<double[]> CollectionWithPoints, string
histLabelX)
        {
            InitializeComponent();
            CreateGraph(zgcGraph, CollectionWithPoints, histLabelX);
            SetSize();
        }
        public new int Width
        {
            get { return zgcGraph.Width; }
            set { zgcGraph.Width = value; }
        }
        public new int Height
        {
            get { return zgcGraph.Height; }
            set { zgcGraph.Height = value; }
        }
```

```
private void CreateGraph(ZedGraphControl zgc, Collection<double[]>
CollectionWithPoints,string histLabelX)
        {
            // get a reference to the GraphPane
            GraphPane myPane = zgc.GraphPane;
            // Set the Titles
            myPane.Title.Text = "Histogram";
            myPane.XAxis.Title.Text = histLabelX;
            myPane.YAxis.Title.Text = "Četnost";
            myPane.XAxis.Scale.MajorStep = 1;

            string R1 = CollectionWithPoints[0].GetValue(0).ToString() + "-" +
CollectionWithPoints[2].GetValue(0).ToString();
            string R2 = CollectionWithPoints[2].GetValue(0).ToString() + "-" +
CollectionWithPoints[4].GetValue(0).ToString();
            string R3 = CollectionWithPoints[4].GetValue(0).ToString() + "-" +
CollectionWithPoints[6].GetValue(0).ToString();
            string R4 = CollectionWithPoints[6].GetValue(0).ToString() + "-" +
CollectionWithPoints[8].GetValue(0).ToString();
            string R5 = CollectionWithPoints[8].GetValue(0).ToString() + "-" +
CollectionWithPoints[10].GetValue(0).ToString();
            string R6 = CollectionWithPoints[10].GetValue(0).ToString() + "-" +
CollectionWithPoints[12].GetValue(0).ToString();
            string R7 = CollectionWithPoints[12].GetValue(0).ToString() + "-" +
CollectionWithPoints[14].GetValue(0).ToString();
            string R8 = CollectionWithPoints[14].GetValue(0).ToString() + "-" +
CollectionWithPoints[16].GetValue(0).ToString();
            string R9 = CollectionWithPoints[16].GetValue(0).ToString() + "-" +
CollectionWithPoints[18].GetValue(0).ToString();
            string R10 = CollectionWithPoints[18].GetValue(0).ToString() + "-" +
CollectionWithPoints[20].GetValue(0).ToString();

            BarItem myCurve1 = myPane.AddBar(histLabelX +" v rozmezí "+ R1 ,
CollectionWithPoints[0], CollectionWithPoints[1], Color.Blue);
            BarItem myCurve2 = myPane.AddBar(histLabelX + " v rozmezí " + R2,
CollectionWithPoints[2], CollectionWithPoints[3], Color.BlueViolet);
            BarItem myCurve3 = myPane.AddBar(histLabelX + " v rozmezí " + R3,
CollectionWithPoints[4], CollectionWithPoints[5], Color.Green);
            BarItem myCurve4 = myPane.AddBar(histLabelX + " v rozmezí " + R4,
CollectionWithPoints[6], CollectionWithPoints[7], Color.GreenYellow);
            BarItem myCurve5 = myPane.AddBar(histLabelX + " v rozmezí " + R5,
CollectionWithPoints[8], CollectionWithPoints[9], Color.Yellow);
            BarItem myCurve6 = myPane.AddBar(histLabelX + " v rozmezí " + R6,
CollectionWithPoints[10], CollectionWithPoints[11], Color.LightYellow);
            BarItem myCurve7 = myPane.AddBar(histLabelX + " v rozmezí " + R7,
CollectionWithPoints[12], CollectionWithPoints[13], Color.Orange);
            BarItem myCurve8 = myPane.AddBar(histLabelX + " v rozmezí " + R8,
CollectionWithPoints[14], CollectionWithPoints[15], Color.DarkOrange);
            BarItem myCurve9 = myPane.AddBar(histLabelX + " v rozmezí " + R9,
CollectionWithPoints[16], CollectionWithPoints[17], Color.Red);
            BarItem myCurve10 = myPane.AddBar(histLabelX + " v rozmezí " + R10,
CollectionWithPoints[18], CollectionWithPoints[19], Color.DarkRed);

            zgc.AxisChange();
        }
```

```
private void SetSize()
        {
            zgcGraph.Location = new System.Drawing.Point(10, 10);
            // Leave a small margin around the outside of the control
            zgcGraph.Size = new System.Drawing.Size((int)this.Width - 20,
                                (int)this.Height - 20);
        }

    }
}
```